VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza medicínských snímků za účelem registrace – vytvoření kvalitního snímku pro následné vyhodnocení

Analysis of the Medical Images – Creating High Quality Image for Consecutive Evaluation

2013 Radek Kuzník

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Radek Kuzník**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Analýza medicínských snímků za účelem registrace – vytvoření kvalitního snímku pro následné vyhodnocení.
Analysis of the Medical Images – Creating High Quality Image for Consecutive Evaluation.

Zásady pro vypracování:

Cílem práce je návrh a realizace modulu systému FOTOM, který umožní fúzi několika medicinských snímků (ultrazvukové případně i endoskopické) se stejnými objekty a vytvoření jednoho kvalitního snímku metodou kombinování snímků (image stitching).

1. Nastudujte základní postupy vhodné ke zpracování uvedených typů snímků.
2. Seznamte se s prostředním NetBeans a programovacím jazykem JAVA.
3. Seznamte se s fotogrammetrickým systémem FOTOMNG.
4. Implementujte požadovanou funkcionalitu pomocí jazyka JAVA a začleňte ji do tohoto systému.
5. Funkčnost demonstrujte na poskytnutých snímcích.
6. Zpracujte programátorskou i uživatelskou příručku a zhodnoťte dosažené výsledky.

Seznam doporučené odborné literatury:

[1] GONZALEZ, C. Rafael; WOODS, E. Richard. Digital Image Processing, 3rd Edition. 2008. 954 str. ISBN 978-0131687288.
[2] Russ, C. John. The Image Processing Handbook, 5th Edition. 2007. 817 str. ISBN 0-8493-7254-2
[3] Ličev, Lačezar: Analýza, modelování, rozpoznávání a vizualizace procesu měření objektů na snímcích, 128 str., Knihy vydané prostřednictvím www.vydejteknihu.cz, Computer Press, a.s., ISBN 978-80-2513-296-8, EAN 9788025132968
[4] Y. Daniel Liang, Introduction to Java Programming, Comprehensive, Prentice Hall; 8 edition, 2010, ISBN-13: 978-0132130806
[5] Sojka, E.: Digitální zpracování obrazu, skripta FEI VŠB - TUO, 1999, ISBN-13: 978-0132130806

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Lačezar Ličev, CSc.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. května 2013

.....

podpis

Poděkování

Rád bych poděkoval vedoucímu diplomové práce doc. Ing. Lačezaru Ličevovi, CSc. za pomoc a rady při tvorbě této diplomové práce.

Dále děkuji své rodině za psychickou podporu a porozumění, které mi velice pomáhalo nejen v době realizace diplomové práce, ale i po celou dobu studií.

Abstrakt

Cílem práce je vytvoření modulu do systému FOTOM, který umožní fúzi několika ultrazvukových či endoskopických medicínských snímků se stejnými objekty a vytvoření jednoho kvalitního snímku metodou kombinování snímků (image stitching). Snímky bude možné spojovat dle parametrů nebo na základě referenčních snímků. Součástí modulů je korekce 2D objektů systému Fotom NG.

Klíčová slova

Digitální zpracování obrazu, Registrace obrazu, Fázová korelace, prahování, Fourierova transformace, FOTOM 2008, FOTOM NG, Netbeans platform, Java

Abstract

The aim of work was create a new module in to system Fotom, which will merge several ultrasound or endoscopic medical images of the same objects and the creation of a high-quality image by combining images (image stitching). Images can be connected according to the parameters or based on reference frames. A part of Module is a correction of 2D objects in Fotom NG.

Keywords

Digital image processing, Image registration, Phase Correlation, Thresholding, Fourier transformation, FOTOM 2008, FOTOM NG, Netbeans platform, Java

Seznam použitých zkratek a symbolů

API	– Application Programming Interface
CT	– Computed Tomography
EBT	– Electron beam tomography
EDT	– Event Dispatch Thread
GUI	– Graphical User Interface
IDE	– Integrated Development Environment
JAI	– Java Advanced Imaging

Obsah

Seznam obrázků

Seznam ukázek

1. Úvod

Medicína je téměř stará jako lidstvo samo, první zmínky se datují již od 5 století před naším letopočtem řeckým lékařem Hippokratem. Jedná se o vědní obor, jehož hlavním cílem je chránit a léčit lidské zdraví. V dnešní době za pomocí moderních technologií je snazší nalézt příčinu, kořen nemoci a cestu k jejímu odstranění. Informační technologie nabízejí lékařskému odvětví mnoho užitečných nástrojů například pro zpracování lékařských snímků získaných z měřících přístrojů. Aplikace zaměřené na medicínu, které analyzují lékařské snímky a následně provádějí korekci, jsou výbornými pomocníky lékařů. Jedním z takových nástrojů je fotogrammetrický systém Fotom, který je již řadu let vyvíjen na katedře informatiky VŠB-TUO Ostrava.

Cílem mé diplomové práce je vývoj nového modulu pro kombinaci lékařských snímků z videozáznamu nebo ultrazvukových či endoskopických snímků z vyšetření pro pozdější zpracování a provedení korekce 2D objektů ve snímku. Vytvořený modul bude následně přidán do nejnovější verze systému FOTOM.

Diplomová práce se dělí na pět hlavních kapitol, které se dále člení na podkapitoly. První kapitola popisuje fotogrammetrii jako vědní obor sloužící v lékařství k vyšetření tkání neinvazivní metodou. Využívá se především ultrazvuková sonda, která registruje odraz ultrazvuku od tkání. V další kapitole je popsáno digitální zpracování obrazu a jeho využití při tvorbě této diplomové práce. Je zde vysvětlena metoda kombinování snímků, která slouží k vytvoření jednoho kvalitního snímku nebo snímku s větším rozlišení.

Ve čtvrté kapitole nalezneme vývoj systému Fotom od jeho prvotní verze až do jeho nejnovější podoby a zjistíme, co vedlo k vytvoření systému Fotom a jaké metody se nejčastěji využívají u vyšetření krční tepny. Následující kapitola se zaměřuje na software prostředky a jejich popis. Jedná se o vývojové prostředí Netbeans platform s Netbeans IDE, knihovnu pro rychlejší tvorbu aplikace u zpracování obrazu jakou je Java Advanced Imaging a využití video knihovny Xuggler pomocí niž lze kódovat a dekódovat video.

V páté kapitole jsou popsány požadavky na systém, návrh, implementace modulu a jeho testování. Kapitola je rozdělena na tři části. První část popisuje metodu kombinací snímků, druhá část se věnuje korekci 2D objektu systému Fotom NG v lékařském snímku a ve třetí kapitole jsou zobrazeny výsledky testování předchozích dvou části kapitoly. Součástí diplomové práce je uživatelská a programátorská příručka.

2. Fotogrammetrie

Fotogrammetrie (z řečtiny: Fotos-světlo, Gramma-písmeno, Metrie-měření) je měření na záznamu pořízeného pomocí světla. Název tohoto vědního oboru zavedl Albrechtem Meydenbauer v roce 1858 [1]. Byl průkopníkem fotogrammetrické dokumentace historických stavebních objektů. Dřívější název fotogrammetrie byl Metrotopografie. Vědní obor fotogrammetrie se zabývá získáváním informací z měřících snímků, získaných pomocí speciálních měřících zařízení. Získané informace určují geometrické vztahy (poloha, velikost, tvar). Při měření není nutný přímý kontakt s předmětem měření, čímž napomáhá měření v těžko dostupných místech. Pomocí projektové geometrie je možné objekt z měření zrekonstruovat, určit mu správnou velikost a tvar.

Fotogrammetrii lze rozdělit do několika kategorií a podkategorií podle dělení a využití. Prvním dělením je podle polohy měřicího přístroje, a to na pozemní, kdy je měřicí přístroj umístěn na nepohyblivém stativu, dále na leteckou, kdy se přístroj nachází na létacím zařízení. Snímky většinou slouží pro tvorbu kartografických map. Posledním typem jsou družicové. Družicové snímky mají vysoké rozlišení a slouží jako geografické mapy či tématické mapy.

Dalším dělením je způsob, jakým je měření na záznam uloženo. Na počátku vzniku byla měření zachytávaná na fotografický materiál s vysokou citlivostí. V dnešní moderní době jsou měření ukládána digitální kamerou skrze CCD snímač.

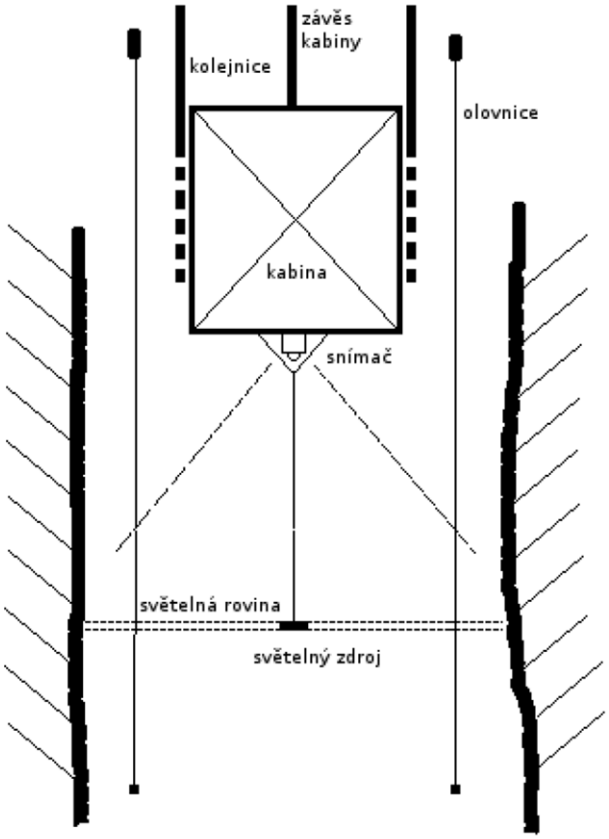
Fotogrammetrii může být jednosnímková, dvousnímková a vícesnímková. Při jednosnímkovém měření jsou získána 2D data z rovinných objektů a terénu se souřadnicemi X a Y. U dvousnímkového (Stereofotogrammetrie) a vícesnímkového měření je možné objekt prostorově zrekonstruovat, protože jsou získána 3D data se souřadnicemi X, Y a Z.

Fotogrammetrie má širokou oblast využití. Jedním z možných jsou podkladové mapy pro stavební projekty spolu s dokumentací staveb, v lesnictví prostorové mapy a mapy těžby dřeva, ve vodohospodářství modelování povodí či oblast záplav.

Fotogrammetrie umožňuje navíc i sledování stavu pomocí vícesnímkového měření. Tato vlastnost se především hodí, v případě chceme-li sledovat stav a vývoj námi vybraného objektu v čase. Vlastnosti pozorování objektu v čase využívá systém Fotom, který tyto informace z měření dále zpracovává a vyhodnocuje. Systém Fotom zpracovává snímky z fotogrammetrie hornictví a lékařství.

2.1 Fotogrammetrie v hornictví

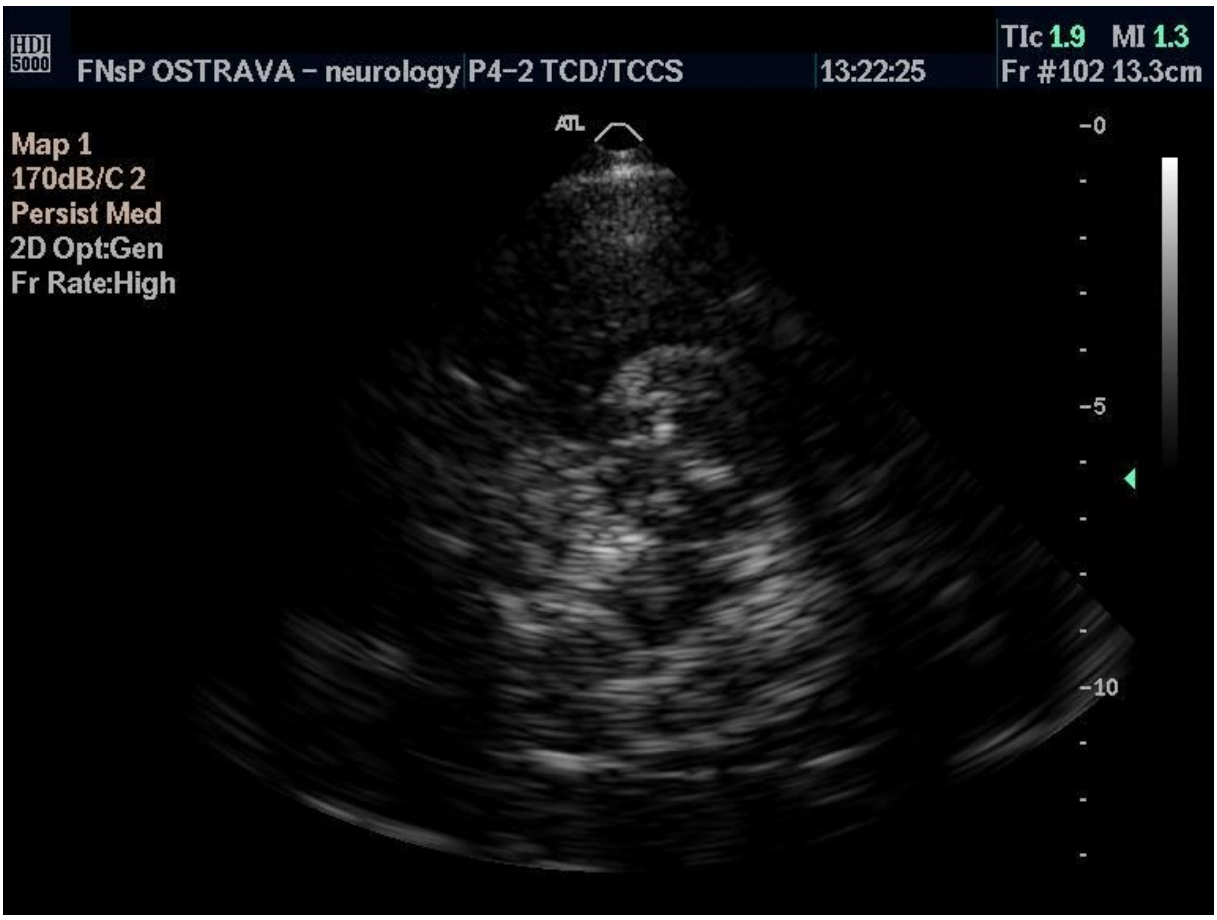
Jak již bylo zmíněno na konci kapitoly o fotogrammetrii, fotogrammetrii lze využit i v hornictví. Měřicí přístroj je připojen ke spodní části důlní kabiny spolu se světelným zdrojem, který vytváří světelnou rovinu. Při pohybu důlní kabiny je světelná rovina snímána v určitých hloubkách (obrázek 1). Tímto snímáním se vytvoří série po sobě jdoucích snímků s informací o hloubce, v jaké byl snímek pořízen. Za pomocí olovnic umístěných blízko důlní kabiny a informaci o hloubce lze určit měřítko pro přepočet souřadnicových hodnot z reálného měření do souřadnicového systému hodnot měřicího snímku. U měření pomocí olovnic však může dojit k jejich rozpohybování, které může vést k nepřesnému přepočtu. Byla vynalezena i jiná metoda, která tuto chybu již neobsahuje, a to tak, že namísto olovnice je přidán další světelný zdroj vytvářející světelnou rovinu. Vzdálenost mezi světelnými zdroji je pevně dána což umožňuje přepočet hodnot mezi souřadnicovými systémy. Hodnoty získané z měření umožňují určit stav jámy, a zda dochází v nějakých místech k jejímu rozšiřování či zužování, čímž se předejde možnému zhroucení důlní jámy.



Obrázek 1 - Fotogrammetrie v hornictví

2.2 Fotogrammetrie v medicíně

Tak, jako v hornictví, i v lékařství je možné využívat fotogrammetrická měření. Snímky jsou pořízeny pro tvrdé tkáně z počítačové tomografie a měkké tkáně z ultrazvuku (Obrázek 2). Získané snímky obsahují informaci o datu a času pořízení, místě pořízení snímku a specifikaci snímku. Za pomocí hodnot dat a času lze sledovat stav a rozvoj nemoci u pacienta v časovém sledu.



Obrázek 2 - Fotogrammetrie v medicíně

3. Digitální zpracování obrazu

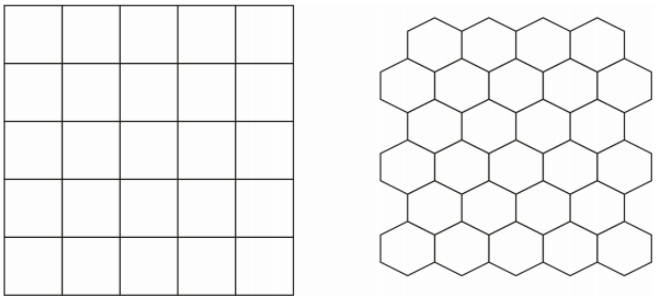
Zdrojem čerpaných informací pro kapitoly 3.2 až 3.8 je literární zdroj [13].

3.1 Definice obrazu

Mluvíme-li o reálném obrazu, mluvíme o obrazu jako o vícerozměrném signálu. Tento reálný obraz lze definovat jako dvourozměrnou funkci $f = f(x, y)$, kde x a y jsou souřadnice v rovině a amplituda f nazveme intenzitou pro každou dvojici souřadnic (x, y) [9]. Obraz v počítači je reprezentován diskrétně, aby bylo možné obraz digitálně zpracovávat, je zapotřebí jej digitalizovat, což znamená převést obraz ze spojitého prostoru do diskrétního, a to za pomocí metod vzorkování a kvantování. Digitální obraz je složen z konečného počtu prvků, které mají konkrétní hodnotu a polohu. Mluvíme o tzv. axelech, označované taky jako obrazové body. Konkrétními hodnotami se myslí barva v obraze.

Vzorkování

Neboli převod analogového obrazu na digitální, který si můžeme představit jako rozdělení obrazu na čtverce a obraz v tomto čtverci převedeme na jeden digitalizovaný obrazový bod. Vzorkování se provádí pomocí vzorkovací matice, která může být hexagonální nebo čtvercová (Obrázek 3) [10]. Po uspořádání do vzorkovací mřížky pokrývají pixely celý digitalizovaný obraz.



Obrázek 3 - Vzorkovací matice [10]

Při správném vzorkování lze vzorkovaný obraz zrekonstruovat na původní obraz. Nelze-li provést rekonstrukci na původní obraz, došlo ke špatnému vzorkování, a tedy ke znehodnocení obrazu. Nejčastěji chybou při vzorkování je podvzorkování (aliasing), vzorkovací frekvence je nižší než dvojnásobek maximální frekvence funkce obrazu. Prevencí před touto chybou je tzv. Shannonův vzorkovací teorém, který říká: "*Přesná rekonstrukce spojitého, frekvenčně omezeného, signálu z jeho vzorků je možná, tehdy pokud byl vzorkován frekvencí alespoň dvakrát vyšší, než je maximální frekvence rekonstruovaného signálu*" [11].

$$\boldsymbol{f_{vz} \geq 2 f_{max}} \tag{1.1}$$

V praxi se vzorkovací frekvence volí ještě o něco větší než dvojnásobek maximální požadované frekvence a u medicínských snímků se volí vzorkovací frekvence 4-5 násobek maximální frekvence ve spektru [11].

Kvantování

U kvantování se jedná o diskretizaci oboru funkčních hodnot, neboli rozdělení oboru hodnot na intervaly, kde každému z nich je přiřazena jedna zástupná hodnota. Kvantování může být rovnoměrné (uniformní) nebo nerovnoměrné (neuniformní). U rovnoměrného kvantování jsou všechny intervaly stejně veliké na rozdíl od nerovnoměrného, kdy mohou mít intervaly různé velikosti. Pro jednoduchost se většinou používá v digitální technice rovnoměrné kvantování. Binárním kódováním kvantovaného vzorku se přiřadí jednotlivým kvantovacím hladinám binární číslo. Počet kvantovacích hladin $n = 2^b$, při b bitech [12]. Při kvantování dochází ke ztrátě informace. Tato ztráta se označuje jako kvantizační chyba a ve zpracování obrazu se projevuje například u hladkého barevného přechodu s malou změnou gradientu jako náhlý skok barev, (viz Obrázek 4). Faktor, který tuto chybu zesiluje, je lidské vnímání. Oko je citlivé na výskyt hran a vnímá tento přechod jako novou informaci v obraze. Změna gradientu ovlivňuje vnímání přilehlých ploch s konstantním jasem.

Obrázek 4 - Ztráta informace při kvantování obrazu

3.2 Konvoluce

Konvoluce je nejvýznamnější základní matematická operace ve zpracování a analýze digitálního obrazu. Hraje významnou roli při detekci hran a aplikaci obrazových filtrů. Konvoluce jako jednoduchý matematický operátor, produkuje ze dvou funkcí, funkci novou. Konvoluci dvou funkcí *f(x, y)* a *h(x, y)* označujeme symbolem *. U dvourozměrného spojitého prostoru je konvoluce definována vztahem

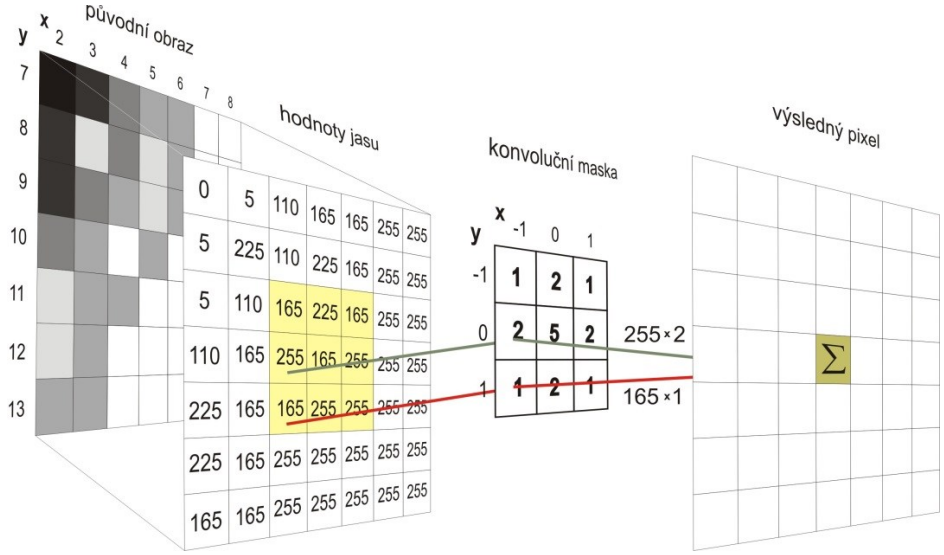
$$f(x,y) * h(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(a,b) * h(x-a, y-b) da db \tag{1.2}$$

Diskrétní dvourozměrná konvoluce

U diskrétní konvoluce pocházejí body z konečného prostoru, definujeme-li tento prostor jako $\Omega = \{(x, y) \mid x = 0,1, .. ,M - 1; y = 0,1, .., N - 1\}$ a nechť signály *f(x, y)* a *h(x, y)* jsou z uvažovaného prostoru, pak platí

$$f(x,y) * h\,(x,y) = \sum_{r=0}^{M-1} \sum_{s=0}^{N-1} f(r,s) h(x-r, y-s) \tag{1.3}$$

U diskrétní konvoluce uvažujme nad funkcí *h* jako na Matici. Matice *h* se označuje jako tzv. konvoluční matice, konvoluční jádro nebo konvoluční filtr. Výpočet probíhá tak, že se matice přiloží na příslušné místo v obraze a každý takto překrytý pixel v obraze vynásobíme hodnotou v příslušném místě matice, čímž provedeme součet všech těchto hodnot. Součet hodnot dává výslednou hodnotu daného pixelu (Obrázek 5).



Obrázek 5 - Diskrétní dvourozměrná konvoluce [12]

Při implementaci diskrétní konvoluce je nutné vyřešit okrajové části obrazové matice, kde některé prvky konvoluční masky se mohou nacházet mimo obrazovou matici při překryvu. Řešením je rozšířit obrazovou matici pomocnými body o poloviční velikost konvoluční masky. Hodnoty pomocných bodů se nejčastěji doplňují hodnotami rovnající nule. Druhou možností je doplnění hodnotami v polohách zrcadlově obrácených dle okraje obrazu. Konvoluce má velký význam u zpracování obrazu pomocí Fourierovy transformace, kdy se využívá konvolučního teorému součinu dvou funkcí (kapitola 3.4).

Konvoluční masky

Návrh a tvorba konvoluční masky je založena na složité matematice a zabrala by příliš mnoho času, proto se využívá především již vytvořené, u kterých je dobře znám výsledek (např. maska pro rozmazání obrazu). Definované masky mívají různé dimenze, pomoci kterých lze nastavit požadované vlastnosti filtru. Nejčastěji se používá matice dimenze 3 x 3, ale můžeme ji narazit i méně využívanou velikost dimenze 9 x 9.

V diplomové práci se setkáváme s obrazy, jež obsahují šum, proto jsem se rozhodl popsat nejčastěji používané filtry k odstranění této chyby v obraze. Nejznámějšími filtry patří Mediánový a Gaussův filtr. Existují i filtry používané k jiným účelům, např. prahování, detekci hran, rozostření a zaostření.

Mediánový filtr

Mediánový filtr patří mezi nelineární filtry, a jak již bylo zmíněno, používá se k odstraňování náhodného šumu v obraze. Jedná se velice účinnou metodu pro potlačení černého nebo bílého šumu. Filtr vezme okolí každého pixelu a z něj vybere medián, který se stává novou hodnotou zpracovávaného pixelu (při dimenzi masky 3 x 3 je medián pátá nejvyšší hodnota prvku).

Gaussův filtr

2D Gaussův filtr je nejpoužívanějším filtrem pro rozostření obrazu a odstranění šumu v obraze. Filtr má podobné výsledky jako prosté průměrování ale na rozdíl od něj využívá konvoluční masky skládající se z hodnot Gaussovy 2D funkce. Tato funkce má tvar

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1.4}$$

Velice důležitým prvkem v rovnici je rozptyl. Pokud bychom zvolili příliš malý rozptyl, je filtrace nevýrazná ale hrany lze stále detekovat, naopak při příliš velkém rozptylu je filtrace výrazná, avšak odpadá možnost správného detekování hran.

Obrázek 6 – Rozostření obrazu pomocí Gaussova filtru

3.3 Prahování

Nejjednodušší metodou segmentace obrazu je proces zvaný prahování, kdy se v obraze detekují celé oblasti. Jedná se o rychlou a jednoduchou metodu, která je velice oblíbená při zpracování obrazu. Hlavní myšlenou prahování je využití faktu, že rozdílné objekty v obraze mají rozdílnou úroveň intenzity (jasu). Jednou z možností realizací prahování je určení mezní hodnoty jasu tzv. práh. Pokud má pixel úroveň jasu vyšší nebo rovný hodnotě prahu je označen za objekt. U pixelu s úrovní jasu nižší než práh označujeme pixel jako pozadí. Výsledek prahování získáváme již po jednom průchodu obrazem, jehož výsledkem je binární obraz, kde nalezené objekty jsou vyznačeny bílou barvou a pozadí černou barvou.

Způsobů prahování je několik, kromě již zmiňovaného globálního prahování s jedním prahem, existuje dále procentní prahování, kde se nebere práh jako hodnota jasu, ale procentuální zastoupení jasové složky. Tento způsob vyžaduje dvojitý průchod obrazem, kdy v prvním kroku jsou spočteny entropie jednotlivých hodnot jasu v obraze a v druhém kroku se určí, zda hodnota jasu pixelu má četnost v obraze větší nebo rovnu hodnotě prahu v případě objektu a nižší v případě pozadí. Tato metoda se využívá při zpracování naskenovaného dokumentu, kdy je průměrné zastoupení textu na stránce okolo 5%.

Upravenou metody globálního prahování je poloprahování, kdy hodnotám jasu vyšší než práh není přidělena hodnota 1 ale jejich původní hodnota. Hodnotám jasu nižší než práh je přiřazena hodnota pozadí. Další metodou je vícestupňové prahování, kdy práh není jenom jeden. Hodnota jasu nacházející se mezi prahy T_i a T_{i+1} určuje, o jaký objekt se jedná. Lze tak určit více objektů najednou. Dále existuje adaptivní prahování, kdy se obraz nejčastěji rozdělí na stejné oblasti čtverců či obdélníků a každé oblastí je určena vlastní hodnota prahu. Metoda je účinná u obrazů s nerovnoměrným osvětlením, kdy do zvolené oblasti nepatří i hranice mezi objekty.

Nejtěžší a nejdůležitější na metodě prahování je správné určení hodnoty prahu. Nejoptimálnějším řešením je, nechat vypočítat hodnotu prahu v obraze automatickou metodou. Tyto metody mohou být založené na analýze histogramu obrazu, kdy histogram obrazu obsahuje jeden nebo více dominantních vrcholů. U obrazů s bimodálním histogramem je vhodné zvolit jako práh minimální hodnotu mezi dvěma vrcholy histogramu. U obrazů s více modálním histogramem je vhodné použít adaptivní prahování tak aby každá oblast obsahovala pouze jeden dominantní vrchol. Posledním uvedeným způsobem nalezení ideální hodnoty prahu je metodou nejmenší chyby. Jedná se o minimalizaci pravděpodobnosti chybného zařazení prvku obrazové funkce (tzn., že prvek obrazu bude chybně vyhodnocen a bude označen jako prvek pozadí obrazu a naopak).

3.4 Fourierova transformace

V kapitole jsou popsány pouze diskrétní případy dvourozměrného prostoru, protože diplomová práce se zabývá a pracuje pouze s daty z diskrétního dvourozměrného prostoru. Fourierova transformace se využívá pro převod z prostorové oblasti do frekvenční a zpátky. Uplatnění nachází v analýze frekvenčního spektra, kdy se Fourierova transformace používá k detekci hran, upravení kvality obrazu, segmentaci obrazu, rekonstrukci obrazu a kompresi obrazu. Obrazové transformace se používají především pro převod obrazu na tvar, který je pro dané zpracování podstatně výhodnější. Označme vstupní obraz *f* a obraz po Fourierově transformaci jako *F*, pak platí

Přímá diskrétní Fourierova transformace

$$F(k,l) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-j2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)}, k = 0,1,\dots,M-1, l = 0,1,\dots,N-1. \quad (1.5)$$

Zpětná diskrétní Fourierova transformace

$$f(m,n) = \frac{1}{\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} F(k,l) e^{j2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)}, m = 0,1,\dots,M-1, n = 0,1,\dots,N-1. \quad (1.6)$$

Konvoluční teorém

"*Fourierova transformace konvoluce funkcí v prostorové doméně je součinem Fourierových obrazů funkcí v doméně frekvenční a naopak součinu v prostorové doméně odpovídá konvoluce v doméně frekvenční. Konvoluce se v tomto případě uvažuje cyklická.*". Platí

$$\mathcal{F}\{f(x,y) * h(x,y)\} = \frac{1}{\sqrt{MN}} F(k,l) G(k,l) \quad (1.7)$$

Teorém posuvu

Dle teorému posuvu můžeme tvrdit, že posun objektu v prostorové doméně vede k fázovému posunu ve frekvenční doméně podle vztahu

$$f(m-m_0, n-n_0) \Leftrightarrow F(k,l)e^{-j2\pi\left(\frac{m_0 k}{M}+\frac{n_0 l}{N}\right)} \tag{1.8}$$

Rychlá Fourierova transformace

Samotná Fourierova transformace nedosahuje příliš dobré časové složitosti (M^2N^2) ve dvourozměrném prostoru, byla proto vymyšlena nová rychlejší Fourierova transformace nazvaná Rychlá Fourierova transformace. Jedná se o metodu, rozděl a panuj, kdy matice N x N je půlena dle lichých a sudých pozic a tento proces pokračuje do doby, kdy zbyde po půlení pouze jeden prvek. Požadavek na použití metody Rychlé Fourierovy transformace vyžaduje čtvercový rozměr u obrazu (tzn., že velikost obrazu musí být N x N) a velikost rozměru musí být rovna násobku dvou. Toho se dá docílit tak, že obraz je rozšířen na nejbližší vyšší násobek dvou, následně zarovnán na střed a doplněn nulovými hodnotami na okrajích, kde již původní obraz nezasahuje.

3.5 Fázová korelace

Fázová korelace je založena na metodě rychlé Fourierově transformaci, jejímž cílem je určit velikost posuvu mezi dvěma částečně se překrývajícími obrazy. Využívá Fourierova teorému o posuvu a znalosti, že dva sobě podobné obrazy mají v jejich křížovém spektru souvislý vrchol právě v místě registrace. O Fázové korelaci můžeme říci, že se jedná o normovanou křížovou korelaci. Metoda je velice oblíbená při registraci obrazu. Kapitola byla převzata z [15]. Máme-li v R^2 definovány funkce $f_1(x,y)$ a $f_2(x,y)$ a dále:

$$\begin{aligned} \mathfrak{F}\{f_1(x,y)\} &= F_1(\omega_x, \omega_y) \\ \mathfrak{F}\{f_2(x,y)\} &= F_2(\omega_x, \omega_y) \end{aligned} \tag{1.9}$$

Kde $\mathfrak{F}$ značí Fourierovou transformaci. Dále předpokládejme vzájemný posuv funkcí:

$$f_2(x,y) = f_1(x+\Delta x, y+\Delta y) \tag{2.0}$$

Pomocí Fourierova teorému o posuvu dostaneme vztah ve frekvenční doméně:

$$F_2(\omega_x, \omega_y) = F_1(\omega_x, \omega_y)\, e^{j(\omega_x \Delta x + \omega_y \Delta y)} \tag{2.1}$$

Dále převedeme funkci F_1 na komplexně sdruženou a získanou funkcí $F_1^*(u, v)$ vynásobíme rovnici:

$$F_2(\omega_x, \omega_y)F_1^*(\omega_x, \omega_y) = e^{j(\omega_x \Delta x + \omega_y \Delta y)} \tag{2.2}$$

Symbol * značí komplexní sdružení. Získaný obraz se nazývá křížové spektrum (cross spectrum) mezi $F_1(u, v)$ a $F_2(u, v)$. Abychom získaly relativní posuv, musíme získat výsledné normalizované křížové spektrum.

$$\frac{F_2(\omega_x, \omega_y)F_1^*(\omega_x, \omega_y)}{\left|F_2(\omega_x, \omega_y)F_1^*(\omega_x, \omega_y)\right|} = e^{j(\omega_x \Delta x + \omega_y \Delta y)} \tag{2.3}$$

Nyní již snadno odvodíme relativní posuv Δx a Δy. Po provedení zpětné Fourierovy transformace získáváme Dirakovu delta funkci se středem v (Δx a Δy):

$$\mathfrak{F}^{-1}\left(\frac{F_2(\omega_x, \omega_y)F_1^*(\omega_x, \omega_y)}{\left|F_2(\omega_x, \omega_y)F_1^*(\omega_x, \omega_y)\right|}\right) = \mathfrak{F}^{-1}\left(e^{j(\omega_x \Delta x + \omega_y \Delta y)}\right) = \delta(\Delta x, \Delta y) \tag{2.4}$$

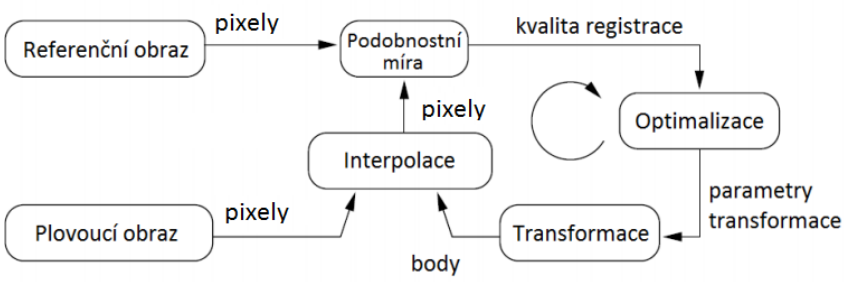
U naznačeného postupu se předpokládají reálné funkce s neomezeným definičním oborem hodnot. Při zpracování obrazu je třeba tento postup aplikovat na diskrétní obrazové funkce konečné velikosti. Řešením tohoto problému je použití diskrétní verze Fourierovi transformace s předpokladem periodického rozšíření obrazových funkcí, kdy Dirakova delta funkce je nahrazena jednotkovým impulsem. Je dokázáno, že i za těchto předpokladů výsledky stále platí [13]. Výsledný vzájemný posun obrázků se zjistí prohledáním křížového výkonového spektra v prostorové oblasti a nalezením maxima, které odpovídá výrazu δ(Δx, Δy) rovnice (2.4).

3.6 Image stitching

Image stitching neboli proces kombinování více obrazových snímků stejné scény s překrývajícími částmi. Proces bývá ve většině případů řešen softwarově, ale existuji již zařízení, kdy celý proces je zpracován hardwarově přímo zařízením, který vytváří výsledný obraz již během pořizování obrazů. Obrazy mohou být pořízeny v různých časech, z různého pohledu nebo snímačů. Tato metoda má široké využití při zpracování obrazu, kdy se metoda využívá pro vytváření obrazu s velkým rozlišením, jakými jsou digitální mapy, satelitní snímky, snímky oblohy nebo vytvoření kvalitního snímku za účelem použití jako referenčního snímku pro další zpracovávání, například u lékařských snímků stavu růst nádoru u pacienta. Kombinaci obrazu můžeme rozdělit na krok registrace a spojení obrazů [2].

3.7 Registrace obrazů

Registrace obrazu umožňuje určit geometrické transformace mezi obrazy. Tyto transformace mohou být způsobeny pohybem pacienta při měření nebo vlivem fyziologické či patologické deformace měkké tkáně ve snímaných obrazech. Registrace určuje míru transformace obrazu, s pomocí níž můžeme zrekonstruovat obraz tak, aby odpovídal referenčnímu obrazu.



Obrázek 7 - Proces registrace, upraveno[3]

Jednou z nejznámějších metod registrace obrazu je fázová korelace. Je založena na Fourierově transformaci a cílem této metody je získat posuv mezi dvěma obrazy, které se částečně překrývají. Fázová korelace má v křížovém výkonovém spektru ostrý vrchol v místě posuvu mezi obrazy.

Metody pro registraci obrazu jsou rozděleny dle různých hledisek a lze je klasifikovat podle možných kritérií:

Dimenze

Metody registrace můžeme dělit na časové, kdy zpracováváme snímky pořízené s časovými rozestupy s cílem sledovat stav nějakého jevu v čase a dále podle dimenze prostoru referenčního obrazu se zdrojovým obrazem (2D/2D, 2D/3D, 3D/3D).

Oblasti lokální a globální transformace

U globální transformace bereme transformaci obrazu jako jeden celek na rozdíl od lokální, kde se na každou část obrazu aplikuje jiný typ transformace s různými parametry.

Typu transformace

Rigidní, neboli tuhá transformace, do nichž patří translace a rotace obrazu.

Afinní, jedná se o transformaci změny měřítka obrazu a tento typ transformace zachovává rovnoběžnost čar.

Projektivní, nezachovává rovnoběžnost čar, ale pouze rovnost čar.

Elastická, někdy označována také jako pružná, deformační nebo nelineární transformace. U tohoto typu transformace může být obraz jakkoliv deformován, ale ve většině případů se jedná o transformaci ne příliš vysokého řádu.

Automatická vs. interakční

Metody registrace mohou být děleny na automatické, poloautomatické a zcela manuální. U manuální metody provádí člověk celou registraci manuálně. Poloautomatická metoda provádí více registračních kroků automaticky, ale závisí na uživateli, aby ověřil správnost výsledku. A u plně automatické metody neumožňuje registrace žádnou interakci s uživatelem a všechny kroky registrace vykonává automaticky.

Domény

Proces registrace může probíhat ve frekvenční nebo prostorové doméně. Prostorové pracují v doméně obrazu hledáním intenzit nebo příznaků v obraze. Metody registrace pracující ve frekvenční doméně nacházejí parametry transformací pro práci v prostorové doméně.

Modality

Jedná se o dělení na multimodální a monomodální. Do monomonálních spadají všechny obrazy, které byly pořízeny stejnou metodou. U multimodálních metod jsou porovnávány snímky pořízené z různých modalit (různými přístroji, různými fyzikálními principy). Multimodální metody jsou více rozšířené, jelikož obsahují detailnější informace a lze jednodušeji určit zájmové objekty.

Pro určení míry úspěšnosti registrace se používají podobnostní míry, které ohodnocují míru podobnosti mezi referenčním snímkem a registrovaným snímkem[3]. Nejznámějšími metodami jsou suma rozdílů čtverců, normovaný korelační koeficient a vzájemná informace.

3.8 Spojení obrazů

Proces spojení obrazů přichází na řadu, jakmile jsou pixely zdrojového obrazu namapovány na referenční obraz a je zapotřební rozhodnout, jak tyto obrazy budou dále zpracovány, aby bylo dosaženo co nejlepšího výsledku. V případě, že proces registrace proběhl správně, provede se pouze spojení obrazů. Avšak u reálných obrazů může dojít k rozmazání, viditelnosti přechodu mezi obrazy nebo zobrazení, tzv. duchů. Pro vytvoření čistého obrazu je zapotřebí určit, které pixely použít a jakou jim dát váhu při procesu spojování obrazů [2]. Nejjednodušší metodou pro vytvoření výsledného obrazu je vzít střední hodnotu každého pixelu z obrazů. Při připojení bodů prvního obrazu k bodám druhého obrazu se postupuje tak, že se vypočítá výřez (hranice) prvního obrazu, kam se druhý obraz bude transformovat. Potom se berou jednotlivé body výřezu prvního obrazu a zpětnou transformací se zjišťuje místo, kam by se promítly do druhého obrazu. Určí se barva tohoto místa a ta se nastaví bodu prvního obrazu. Jas se určuje dvěma způsoby, buď se najde nejbližší skutečný bod (nejbližší soused), nebo se interpoluje barva ze čtyř nejbližších sousedů [17]. Pokud označíme jasovou funkci prvního obrazu jako I a podobně J u druhého obrazu, bude platit pro:

1. Metodu nejbližšího souseda

$$I(u,v) = J(round(H^{-1}(u,v))) \tag{2.5}$$

2. Bilineární interpolaci

$$\begin{aligned} I(u+\alpha, v+\beta) = (1-\beta)\big((1-\alpha)J(u,v) + \alpha J(u+1,v)\big) \\ + \beta((1-\alpha)J(u,v+1) + \alpha J(u+1,v+1)) \end{aligned} \tag{2.6}$$

, kde $(x+\alpha, y+\beta) = H^{-1}(u,v),\ \alpha,\beta \in (0,1);\ x,y \in N$

4. Systém Fotom a medicínské snímky

4.1 Fotom 2008

Systém Fotom je na katedře FEI VŠB-TUO vytvářen od roku 2001. Jedná se o systém pro zpracování obrazu. Tento systém byl zpočátku navržen a použit pro měření důlních jam, avšak jako každý systém i systém Fotom byl postupně vylepšován a obohacován o další funkčnost. Tento systém nesl název Fotom 2008 [4] a byl vyvíjen v jazyku C++. Systém Fotom 2008 obsahoval tyto moduly:

- Modul FOTOM1 – označení zájmových bodů a objektů,
- Modul FOTOM2 – 2D modelovaní procesu měření, měření odchylek a porovnání dvou měřen,
- Modul FOTOM3 – 3D modelovaní procesu měření,
- Modul FOTOM4 – Animace procesu měření,
- Modul FOTOM5 – Rozpoznávání zájmových bodů a objektů

Jednalo se o velice silný nástroj pro analýzu a zpracování důlních snímků. Systém Fotom byl navržen jako jednoúčelový systém a nepředpokládalo se další rozšiřování. Protože Fotom 2008 nebyl navržen jako modulární, všechny nově vznikající moduly byly vytvořeny jako samostatné programy, které byly následně vloženy přímou úpravou zdrojového kódu systému Fotom. Jelikož, každé přidání nové funkčnosti do systému Fotom stalo víc a víc času a veškeré úpravy byly čím dál složitější, jelikož i nově přidaná funkce mohla ovlivnit, již fungující metody, bylo zapotřebí vydat se jinou cestou, kdy by byly plně využity moderní nástroje a rozdělit aplikaci na jednotlivé moduly s jednoduchou možností přidání nových funkčností skrze moduly, které by byly na sobě nezávislé, a přesto mohly využívat jakýkoliv modul, pokud to situace vyžadovala. Vznikl tedy nový systém Fotom NG (Nové generace).

4.2 Fotom NG

Hlavním požadavkem na nový systém byla modulárnost tak, aby byl systém vyvíjen moderními technologiemi a jakákoliv úprava či přidání modulu bylo co nejjednodušší. Tohoto úkolu se zhostili tehdejší diplomanti Ing. Lukáš Krahulec a Ing. Jan Král.

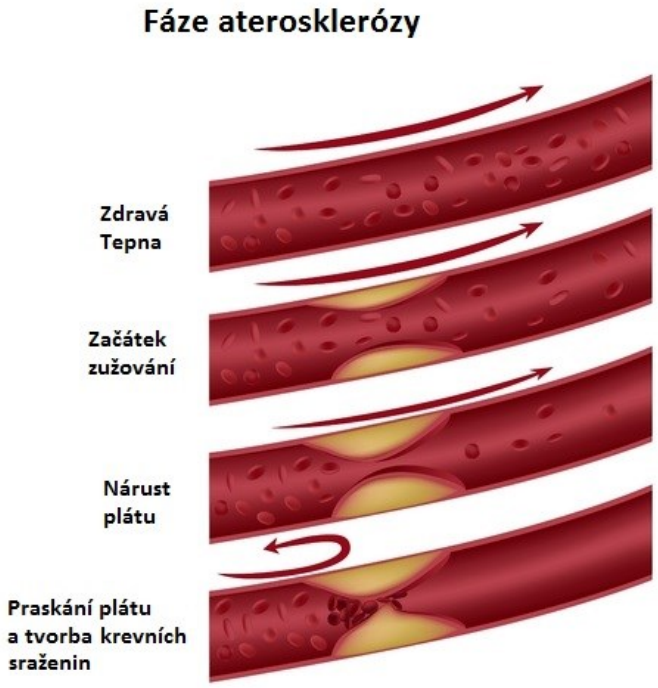
Systém Fotom NG byl vyvíjen v programovacím jazyku Java ve vývojovém prostředí Netbeans platform, který umožňoval již zmiňovanou modularitu. Nový systém obsahuje jednotné API, které může využít každý nově přidaný modul bez nutnosti zásahu do zdrojového kódu modulu API. Ze systému Fotom 2008 byla většina funkcionality byla převedena do této nové verze.

Fotom NG byl v následujících letech 2011 až 2013 rozšířen díky bakaláři Tomášovi Hudečkovi, diplomantovi Ing. Petrovi Zajíci a Ing. Tomášovi Pytlíku. Tito pánové rozšířily funkčnost Fotomu NG o další funkce pro zpracování obrazu.

4.3 Vyšetření krční tepny

Systém Fotom NG a tato diplomová práce pracuje především se snímky krční tepny, z tohoto důvodu bych rád napsal o tom, jak vyšetření krční tepny probíhá, na jaká úskalí můžeme narazit u během vyšetření a proč je dobré vyšetření provádět.

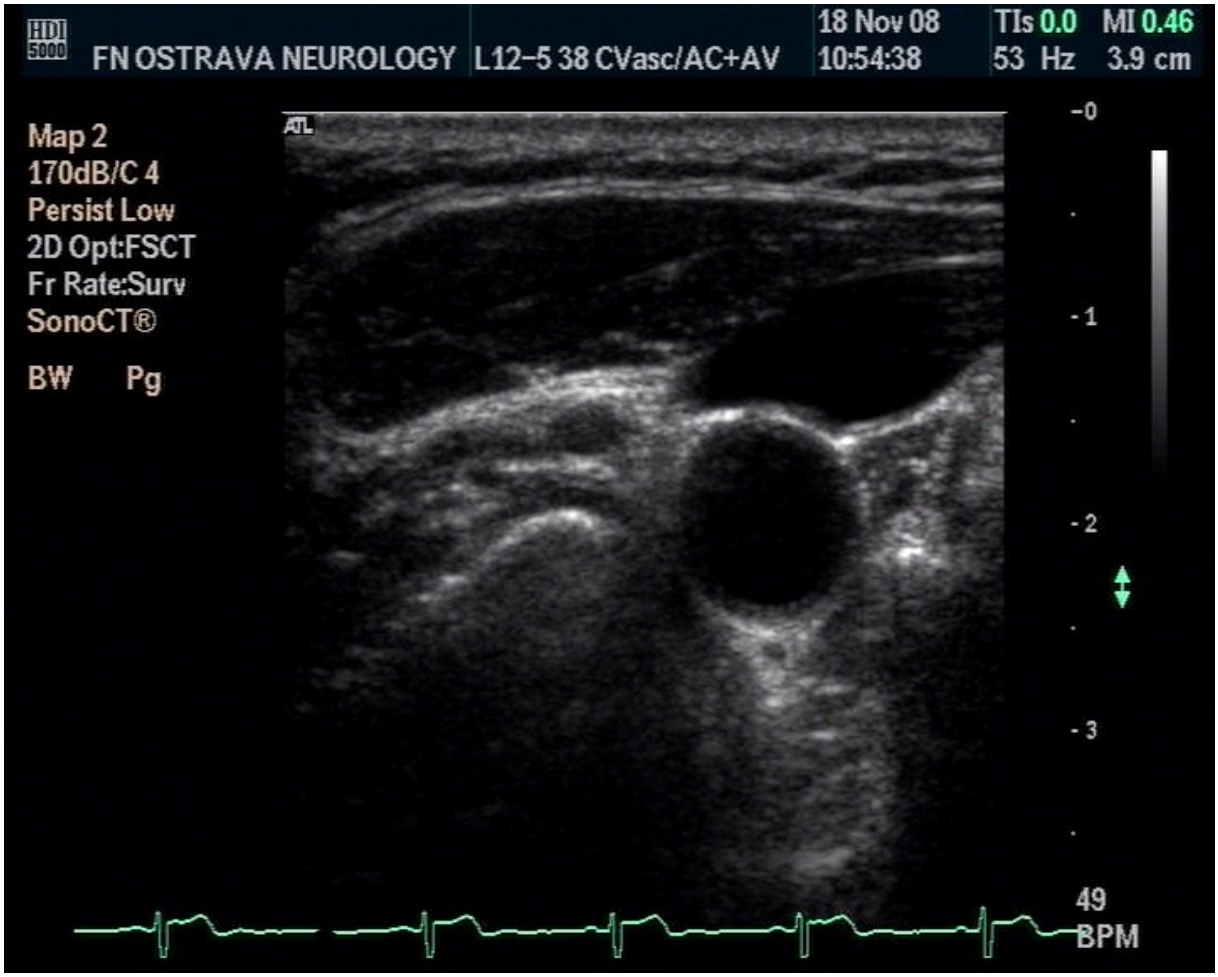
Vyšetření krční tepny se provádí při prevenci a podezření na aterosklerotické zúžení krční tepny ukládáním tukových látek do jejich stěn, jedná se o tzv. kornatění. Toto zužování tepny nemusí mít delší dobu žádný projev. Jedním z projevů tohoto onemocnění je dehydratace, pokles krevního tlaku, které způsobuje motání hlavy a únavu. Nejhorším případem projevu je v případě, pokud krční tepna je zúžením menší, než 65% původní velikost tepny, pak nastává mozková mrtvice, kdy se mění tok krve v zúžené části krkavice a dochází k větší tvorbě krevních sraženin (Obrázek 8). Tato sraženina se může dostat z krkavice do mozku a způsobit zmíněnou mozkovou mrtvici. Ateroskleróza je nejčastější příčinou úmrtí dospělého člověka ve vyspělých zemích.



Obrázek 8 - Fáze aterosklerózy

Při prevenci nebo po mozkové mrtvici se provádí vyšetření nenáročnou a neinvazivní metodou. K tomuto vyšetření se používá ultrazvuk, který využívá lékařská diagnostická zobrazovací technika zvaná sonografie. Diagnostická informace je získána zachycením, zpracováním a zobrazením ultrazvukových signálů, odražených od tkáňových rozhraní. Pro vyšetření krční tepny se diagnostika nazývá sono karotid, kdy je skrze lineární ultrazvukovou sondu do těla pacienta vysíláno ultrazvukové

vlnění o rozsahu 5 až 12 MHz. Takto vyslaná vlnění pronikají hmotou, kde na rozhraní dvou částí s odlišnými vlastnostmi dochází k odražení části vln, jenž se vracejí zpátky ke zdroji. Odražené vlny a jejich časové zpoždění zpracuje diagnostický sonograficky přístroj a zobrazí černobílý obraz na obrazovce [7], kde kapaliny jsou označeny černou barvu a pevné prvky bílou (Obrázek 9). Pokud bychom chtěli zobrazit i rychlost proudění krve v tepně, lze využít dopplerovské ultrasonografie. Důležité u této metody je, že zjištěná rychlost není rychlost toku, ale složek rychlosti ve směru od sondy nebo k sondě.



Obrázek 9 - B-obraz krční tepny z FN Ostrava

5. Softwarové prostředky

5.1 Netbeans platform

Při vývoji v programovacím jazyku Java, můžeme využít nepřeberného množství editorů zdrojového kódu, ale pouze ty lepší z nich jsou schopné zvýrazňovat syntaxe, spouštět zdrojový kód a krokovat jej. Mezi nejznámější editory zdrojového kódu Java patří programy Eclipse a Netbeans. Netbeans nabízí svou platformu zvanou Netbeans Platform, pro snadnější tvorbu uživatelsky přívětivých aplikaci. Netbeans Platform je především frameworkem, nabízející velké množství prvků pro tvorbu aplikací a umožňuje tvorbu uživatelského rozhraní skrze framework swing.

Stavebním kamenem Netbeans Platform jsou logicky oddělené částí aplikace, tzv. moduly. Moduly jsou dynamicky načítány pomocí runtime containeru. Každý modul má vlastní soubor manifest obsahující informace o modulu a závislostech, soubor Layer.xml, ve kterém jsou definovány akce, menu, služby, komponenty a palety nástrojů. Od verze Netbeans Platform 7 je možné všechny tyto věci již nadefinovat přímo ve třídách pomocí anotací. Ukázka definice ve verzi 7:

```
1.  @TopComponent.Description(preferredID = "MyViewerTopComponent",
2.      persistenceType = TopComponent.PERSISTENCE_ALWAYS)
3.  @TopComponent.Registration( mode = "explorer", openAtStartup = true)
4.  @ActionID( category = "Window",  id = "org.myorg.myviewer.MyViewerTopComponent")
5.  @ActionReference( path = "Menu/Window" /*, position = 333 */)
6.  @TopComponent.OpenActionRegistration( displayName = "#CTL_MyViewerAction",
7.      preferredID = "MyViewerTopComponent")
```

Ukázka 1 - Registrace vlastnosti TopComponenty ve verzi 7

Všechny lokální soubory Layer.xml jsou při spuštění spojeny do jednoho souboru a vytváří tak stromovou strukturu. Kromě již dvou zmíněných souborů manifest a Layer může modul obsahovat i třídy a zdroje jako jsou ikony, obrázky, lokalizace atd. Netbeans Platforma nabízí nepřeberné množství API, mezi hlavní patří [5]:

Windows Systém API

API je zodpovědné za zobrazení a správu všech oken aplikace. Pomocí Windows System API je možné vytvořit okno aplikace, jenž dědí ze třídy TopComponent. Po vytvoření okna aplikace uživatel získá čisté plátno, které si dále může jakkoliv upravovat přidáváním grafických prvků a funkčností. Takto vytvořená okna je možné dokovat pomocí definovaného módu. Módy je možné vytvářet a upravovat. Základními mody jsou Explorer, Editor, Output a Navigator. Tyto mody určují, na jakém místě se bude defaultně zobrazovat okno aplikace.

Nodes API

Nodes API se stará o prezentaci dat. S tímto API je úzce svázáno Explorer API, které je zodpovědné za zobrazování a správu Uzlů (Node). Uzly se využívají pro přenos dat do uživatelského rozhraní aplikace, kterého mohou využít funkce a uživatelské akce. Data pro Uzly jsou získávána z různých zdrojů, ale především jde o mapování souborového systému do datových objektů zvaných Uzly. Jaký typ objektu Uzlu bude vytvořen, záleží na definovaných filtrech.

Action API

Jedná se o API určené pro akce mezi uživatelskými komponentami. Po vytvoření jakékoliv akce je potřeba akci registrovat v souboru Layer.xml. Akcím lze přiřazovat klávesové zkratky a mohou být jednoduše přidány do menu.

Lookup API

Krásným příkladem využití této API knihovny je objekt, k němuž chceme mít přístup z jakéhokoliv modulu v aplikaci. Takový objekt je vložen do Map struktury, kde klíčem je třída a hodnotou je její instance. Tyto vložené objekty je možné sdílet skrze celou aplikaci, byl-li vložen do globálního pokupu. Avšak existuje i lokální lookup, který je vázán pouze na aktuální kontext a jeho změny.

5.2 Java Advanced Imaging

Pro rychlejší a snazší vývoj modulů pro systém Fotom byla vybraná knihovna Java Advanced Imaging (JAI). Tato knihovna je do systému Fotom integrována od verze Fotom NG a slouží jako pomocník při zpracování obrazů. Většina nástrojů, které knihovna JAI nabízí je implementováno v modulech Fotom NG. Z důvodu, že je již tento nástroj plně využíván a implementován ve Fotomu, rozhodl jsem se tuto knihovnu také využít.

5.3 Xuggler

Poslední verze systému Fotom, Fotom NG má díky Ing. Petrovi Zajícovi implementovanou knihovnu Xuggler pro zpracování videa [6]. Tato Java knihovna Xuggler poskytuje jednoduché API pro práci s médii a využívá knihovnu FFmpeg pro dekompresi a kompresi média. Poskytované API nabízí dva nástroje API. Prvním z nich je **MediaTools API,** což je velice jednoduché API pro snadné užití při kódování, zobrazování a dekódování videa a zvuku. Druhé API zvané **Advanced API**, je již pokročilejším nástrojem, které nabízí vývojáři širokou paletu nastavení, rozšíření a funkcí.

6. Návrh, implementace a testování

Cílem bylo vytvořit nový modul, který lze snadno začlenit do stávajícího systému Fotom NG tak, aby využíval všechny dostupné prvky systému Fotom NG. Vývoj probíhal v jazyce Java s Netbeans Platform verzi 7 ve vývojového prostředí Netbeans 7.0.1. Tato verze Netbeans byla doporučena vedoucím diplomové práce doc. Ing. Lačezarem Ličevem CSc. jako bezproblémová a kompatibilní pro vývoj nových modulů do systému Fotom NG. Využil jsem již implementované knihovny JAI a Xuggler čímž jsem dosáhl napsání mnohem kratších zdrojových kódů, které jsou srozumitelné a dostatečně rychlé. Pro vývoj bylo nutné nainstalovat na vývojové zařízení zmíněné knihovny a aplikace spolu se spustitelným zdrojovým kódem Fotomu NG.

Vývoj modulu byl rozdělen na dvě logické části. První část se zabývala analýzou a korekcí medicínských snímků s cílem vytvořit kvalitní snímky pro pozdější použití. V druhé části se provádí analýza a korekce geometrických objektů systému Fotom NG ve snímcích.

6.1 Korekce medicínských snímků

6.1.1 Specifikace požadavků

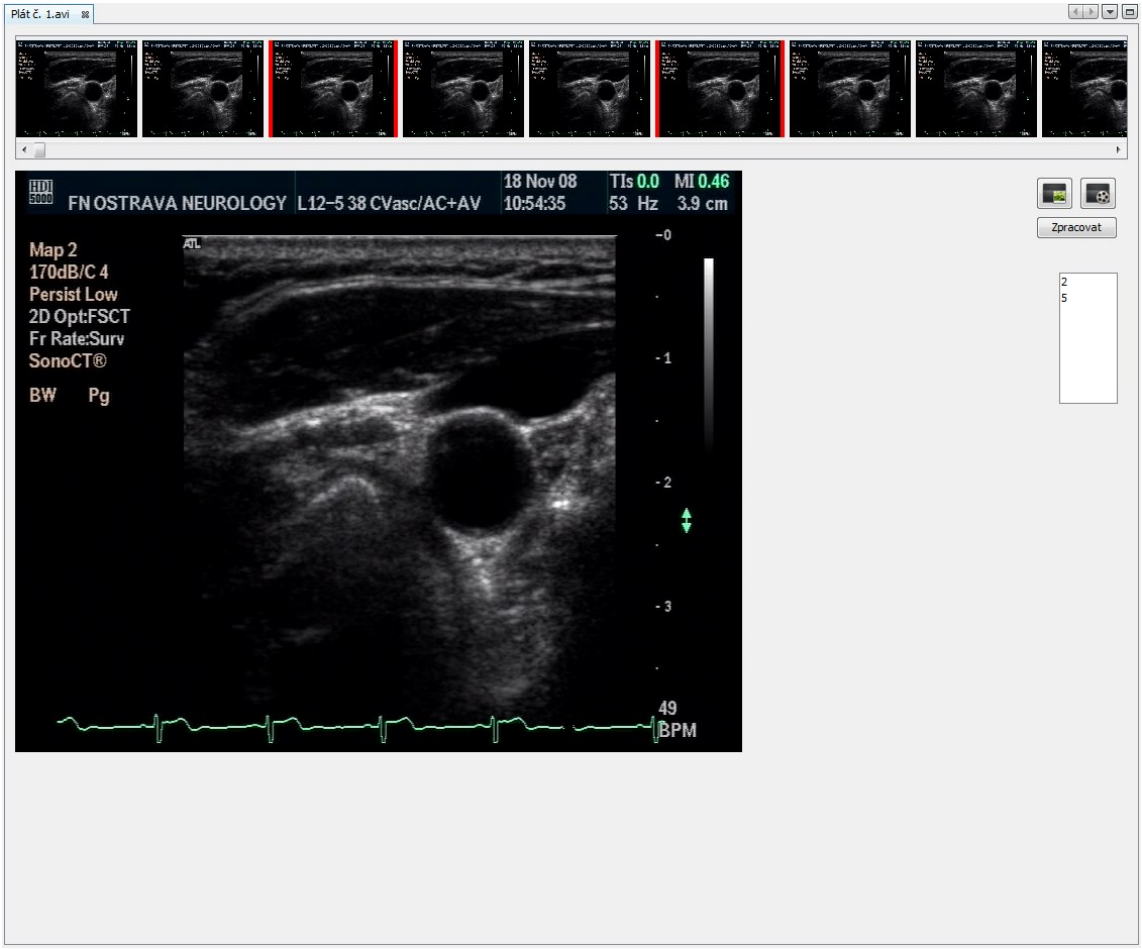
Součástí tvorby nového modulu pro systém Fotom NG byly požadavky zadané vedoucím diplomové práce. Níže jsou vypsány požadavky, které byly zadány pro první část modulu:

- Načíst lékařské snímky a videa různých formátů.
- Načtené video dekódovat na snímky a po korekci snímků umožnit vytvořit ze snímků video.
- Zobrazit všechny načtené snímky na obrazovce spolu s detailním náhledem vybraného snímku ze série.
- Doplnit chybějící částí objektů na snímku definovanou barvou, průměrnými hodnotami všech snímků, hodnotami z referenčního snímku či uživatelsky definovanou barvou.
- Umožnit uživateli zvolit ze série snímků, které mají být použity jako referenční pro korekci snímku.

Takto definované požadavky byly během tvorby diplomové práce blíže specifikovány a detailněji popisovány. Mým cílem bylo vytvořit co nejvíce abstraktní návrh tak, aby například proces analýzy nepoznal, z jaké zdroje jsou dané snímky, a zpracoval tato data bez jakýchkoliv problémů, ať už zdrojem bylo video či sekvence obrazů. Část analýzy a korekci medicínských snímků jsem rozdělil na podčásti. Důvodem tohoto rozdělení je zlepšení orientace v kapitole a ukázka postupného vzniku této části modulu v časovém sledu.

6.1.2 Návrh GUI

Ze specifikace požadavku jsem získal dostačující představu o tom, jaký má mít tento modul GUI vzhled. Dalším krokem bylo vhodně zvolit, které ovládací prvky využijeme a jak budou přidány do stávajícího grafického návrhu Fotomu NG. Systém Fotom používá defaultní architekturu Netbeans platform na bázi dokumentu. Tato architektura definuje rozložení prvků v okně, kde Menu je umístěno v horní části okna, editor uprostřed okna a ve spodní části je umístěn stavový panel. Okno pro korekci medicínských snímků se bude otevírat přes novou položku Korekce obrazu v menu Soubor a bude obsahovat panel pro zobrazení náhledů všech načtených snímků, panel pro detail snímku, list s čísly referenčních snímků vybrané uživatelem a dva tlačítka pro načtení videa a snímků (Obrázek 10).



Obrázek 10 – GUI nového modulu pro korekci snímků

6.1.3 Načítání snímků

Snímky mají dva způsoby, jak mohou být načteny do panelu náhledů, a to z videa nebo obrazů. Pro načtení těchto cest jsem zvolil Swing komponentu *JFileChooser*, pro kterou jsem implementoval třídu *FileFilter*, která umožní filtrování souborů obrázků a videí v adresáři. Uživatel snadněji najde hledaný soubor, aniž by musel projít všechny soubory ve složce. Dále přišlo na řadu, jak navrhnout načtení snímků do panelu náhledu tak aby zmiňovaný panel nevěděl, z jakého zdroje data pochází, a přidání jakéhokoliv nového zdroje nepotřebovalo změnu kódu v panelu náhledů. Rozhodl jsem se proto vytvořit abstraktní třídu *Container*, která se inicializuje v okně aplikace.

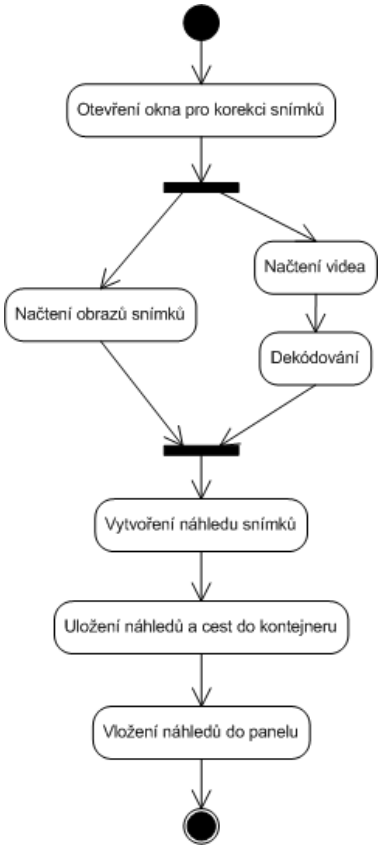
Třída *Container* obsahuje hašovací tabulku *thumbnailsMap* pro náhledy snímků. Klíčem této tabulky je třída *JLabel* a hodnotou je číslo pořadí v mapě. Důvod výběru třídy *JLabel* jako klíče bude vysvětleno ke konci této podkapitoly. Dále je implementován list s čísly vybraných referenčních snímků a abstraktní metody *getImage()* a *getImage(int index),* které budou sloužit k získání detailního snímku dle aktuálního nebo zadaného indexu.

Pro načtení snímků z obrázků jsem vytvořil potomka ze třídy *Container* třídu *ImageContainer*, který má implementované pole obsahující cesty k obrazům se snímky a vlastní implementaci metody *getImage()* a *getImage(int index),* kde je snímek načítán pomocí nativní Java knihovny *javax.imageio.ImageI* jako instance třídy *BufferedImage.* Do konstruktoru třídy *ImageContainer* se vkládá pole cest k obrázkům, které je následně uloženo do privátní proměnné *imagePaths*. Cesty jsou následně procházeny a z nich načteny všechny obrázky se snímky, ze kterých jsou vytvořeny náhledy, které jsou dále ukládány skrze rodičovskou metodu *AddThumbnail(BufferedImage img, int seq)*, kde proměnná *seq* je číslo pořadí vkládané do rodičovské proměnné *thumbnailsMap*. Pokud je potřeba vytáhnout celý obrázek volá se metoda *getImage()* nebo *getImage(int index)*, která najde dle indexu cestu v poli cest a pomocí vybrané cesty vrátí načtený obrázek se snímkem.

Stejně jako pro obrázky byla vytvořená třída dědící ze třídy *Container,* vytvořil jsem i pro video podobnou třídu s názvem *VideoContainer*. Třídy *ImageContainer a VideoContainer* jsou si velice podobné, liší se však především v implementaci metod *getImage()*, *getImage(int index)*. Třída *VideoContainer* již neobsahuje pole cest, ale pouze jednu cestu k video souboru, která se předává v konstruktoru třídy. Aby bylo možné vytvořit náhledy snímků, je potřeba video prvně dekódovat na snímky a z nich následně vytvořit náhledy. K tomu jsem využil již integrované knihovny Xuggler v systému Fotom NG, která se nachází v modulu s názvem *xuggler* [6]. Rozhodl jsem se použít pokročilejší API s názvem Advanced API pro dekódování snímků z videa.

Prvním krokem pro dekódování pomocí Advanced API je vytvořit instanci třídy *IContaineru* a pomocí ní otevřít požadovaný video soubor, čímž získáme informace o počtu proudů ve video souboru. Jelikož každý video soubor se většinou skládá z proudu videa a audia, bylo hlavním cílem vytáhnout právě proud videa typu *IStream*. Takto získaný proud nám poskytl další informace o videu a to celkový počet snímků ve videu, rychlost přehrávání v počtu snímků za sekundu a typ kodéru. Jakmile byly tyto informace získány, mohlo se začít s procházením všech paketů a z nich následně získat jednotlivé snímky z videa. Z takto získaných snímků jsou vytvořeny náhledy, které jsou

následně ukládány pomocí rodičovské metody *AddThumbnail(BufferedImage img, int seq)*. Aby nedocházelo ke stálému znovunačítání, jsou informace o videu uložené do proměnných ve třídě *VideoContainer*.



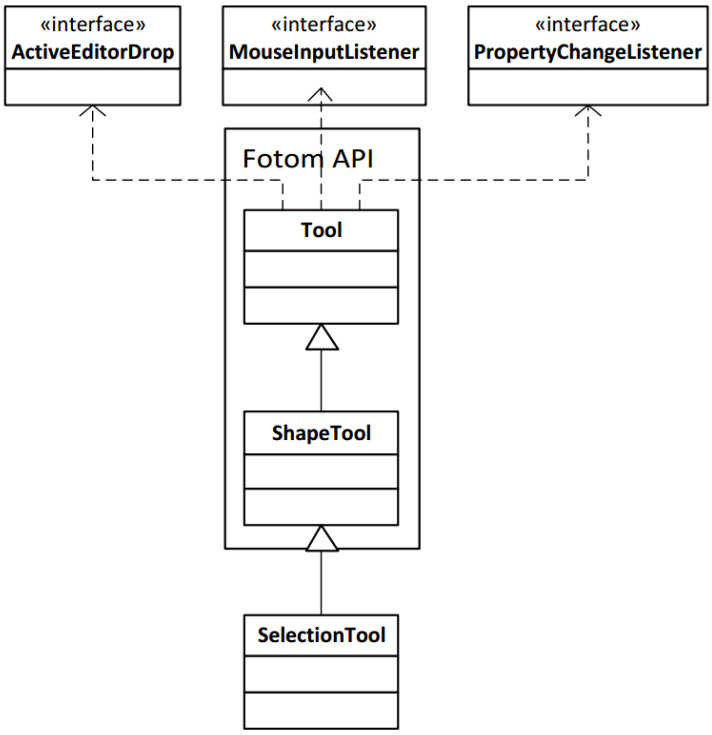
Obrázek 11 - Stavový diagram načtení snímků

K použití metody ukládání cest jsem se rozhodl kvůli menším nárokům na paměť, kdy nejsou všechny nahrané snímky přímo v paměti, ale pouze jejich náhledy, pokud uživatel klikne v panelu na obrázek pro zobrazení detailu, načte se skrze cestu k souboru. Z časové náročnosti, je to o něco pomalejší, ale paměťová náročnost je značně lepší. Použití třídy *JLabel* jako klíče u hašovací tabulky se mi zdálo jako nejlepší způsob, jelikož událost kliknutí na náhled snímků vrací právě instanci třídy *JLabel,* pomocí právě této instance jsem jednoduše zavolal metodu *get*(*Object* key), která mi vrátila požadovaný index pro výběr cesty k souboru.

6.1.4 Nastavení výpočtu

Aby bylo možné výpočet provést, je potřeba určit volitelné a povinné parametry. Jedním z volitelných parametrů je určení, zda výpočet má probíhat ve vztahu s referenčními snímky. Pro referenční snímky byla vytvořena možnost je zvolit z panelu náhledů, kde po dvojitém kliknutí se kolem náhledu vytvoří červený rámeček a navíc se přidá do listu *referenceIndexes* instance třídy *Container*. List *referenceIndexes* obsahuje pouze čísla indexu náhledů v poli. Tento list byl vytvořen z důvodu snazšího odstraňování vybraných referenčních snímků, jinak by uživatel musel procházet celý panel náhledu, a jakmile by uživatel našel hledaný snímek, dvojklikem by jej odznačil.

Dalším volitelným parametrem, který byl vytvořen je *SelectionTool*. Podle tohoto názvu byla vytvořena i třída dědící z *ShapeTool*. Třída *ShapeTool* je abstraktní třída odvozená od obecné třídy *Tool*, reprezentující nástroje pro kreslení na plátno. *ShapeTool* je použit u nástrojů pro definování tvarů na snímku. Potomky třídy *ShapeTool* jsou objekty jako *CircleTool* a *PolygonTool*. Tyto třídy mají v sobě zahrnuty již funkce, díky nimž můžeme získat informace o objektu. Více o třídě *ShapeTool* lze nalézt v [8]. Pro vytvoření třídy *SelectionTool* posloužil jako vzor *SelectionTool* z [6]. Tato třída slouží jako nástroj pro výběr určité oblasti v plátně v mém případě pouze oblast, kterou chci zpracovávat. Nástroj má tvar obdélníku, u nějž je možné různě měnit velikost a pozici v plátně. Pro vytvoření nástroje bylo použito Fotom API, které obsahují všechny implementované nástroje v systému Fotom (Obrázek 12).



Obrázek 12 - Třídní diagram nástroje SelectionTool[6]

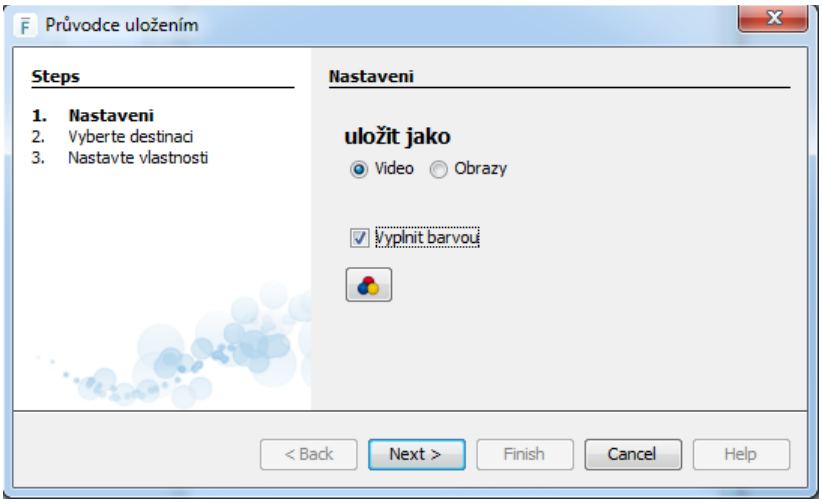
Takto vytvořený nový nástroj je potřeba do aplikace registrovat pomocí souborů layer.xml. Definuje se název nástroje, do jaké skupiny nástrojů má patřit a cesta k popisnému XML souboru, který obsahuje informace, jakou třídu má nástroj volat pro inicializaci, cesty k ikonám nástroje v různých velikostech, název a tooltip (Ukázka 2).

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE editor_palette_item PUBLIC "-//NetBeans//Editor Palette Item 1.1//EN"
  "http://www.netbeans.org/dtds/editor-palette-item-1_1.dtd">

<editor_palette_item version="1.0">
   <class name="org.fotomapp.imagecorrection.SelectionTool" />
   <icon16 urlvalue="org/fotomapp/imagecorrection/resource/SelectionTool.png" />
   <icon32 urlvalue="org/fotomapp/imagecorrection/resource/SelectionTool16.png" />
   <description localizing-bundle="org.fotomapp.imagecorrection.Bundle"
         display-name-key="NAME_selection"
         tooltip-key="HINT_selection" />
</editor_palette_item>
```

Ukázka 2 - Popisný soubor nástroje SelectionTool

Nástroj SelectionTool vznikl, aby nedocházelo ke špatným výpočtům, kdy do výpočtu se zahrnula i okrajová oblast snímku s informacemi o EKG, datem pořízení atd. Druhým důvodem bylo, aby výpočet probíhal co nejrychleji, pomocí výběru menší oblasti, která bude zpracovávána.

Dalším krokem před samotným zpracováním snímků je průvodce *SaveWizard*. Byl navržen pro nastavení typu uložení výsledku korekce snímku, cestu pro uložení výsledku a v neposlední řadě nastavení zda pro vyplnění chybějících části ve snímku, kdy uživatel zvolí, zda použít uživatelsky definovanou barvu (Obrázek 13). Pro tvorbu průvodce byl použit průvodce pro tvorbu průvodce v Netbeans Platform.



Obrázek 13 - Průvodce SaveWizard

6.1.5 Korekce

Nejdůležitější součástí časti modulu, bylo nalézt posun mezi dvěma částečně překrývajícími snímky tak, aby šly snímky následně zarovnat a doplnit jejich chybějící části. Dodané testovací snímky měly mezi sebou posuv v řádu tří až pěti pixelů. Deformace snímků byla na tom podobně, velikost deformace činila dva až čtyři pixely. Rozhodl jsem se využít tedy pro zjištění posuvu Fázovou korelaci.

K tomu mě vedly informace o obrazu, a to, že dodané testovací snímky pořízené pomocí ultrazvuku obsahují informace o pacientovi a údaje získané z ultrazvuku, které jsou umístěny na okrajích lékařského snímku. Aby nebyly okraje započítávaný při výpočtu posunu, vytvořil jsem SelectionTool (Kapitola 6.1.4), pomocí kterého se vybere pouze oblast tepny. U takto vybraných oblastí se jednalo o posun dvou až čtyř pixelu a minimální deformace neměla mít vliv na velikost posuvu. Snímky dále neobsahovaly rotaci či translaci. Jedná se o poměrně rychlou a přesnou metodu pro zjištění posuvu mezi dvěma částečně překrytými obrazy.

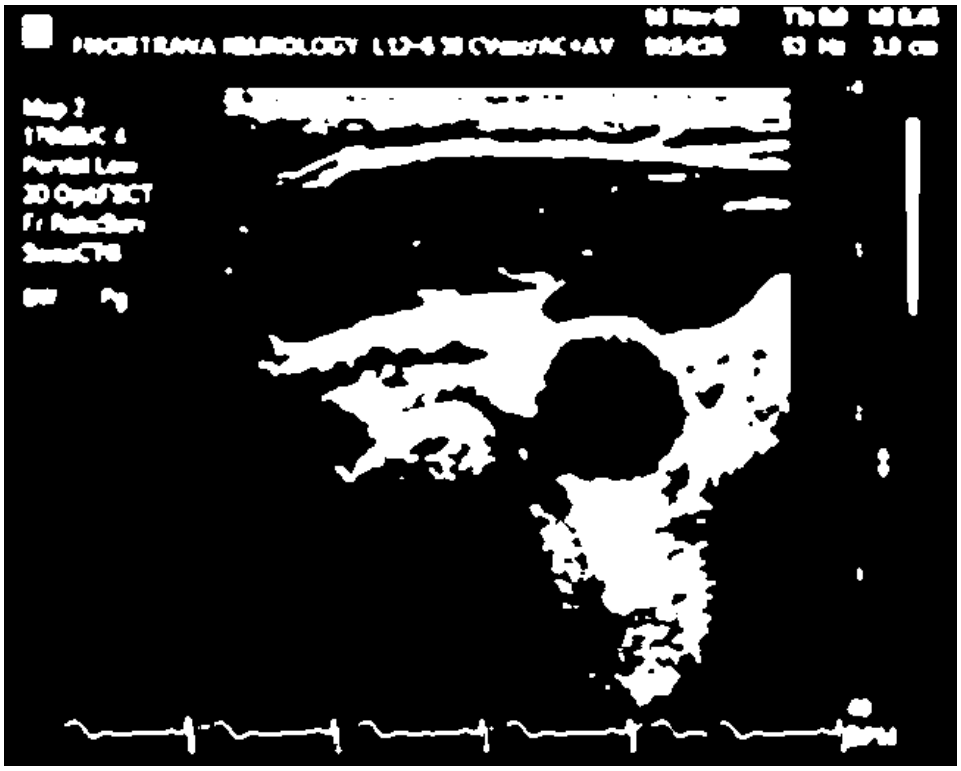
K provedení korekce předchází i krok předzpracování, díky čemuž odstraníme z obrazu nedokonalosti, které by mohly vést k nepřesnému určení posuvu a špatnému vyplnění oblasti. Obrazy obsahovaly defekty, okraje s informacemi o pacientovi a měření. Ty lze odstranit pomocí nástroje *SelectionTool*.

Odstranění šumu

Medicínské snímky pořízené pomocí ultrazvuku (Obrázek 9) často obsahují velké množství šumu, k jeho odstranění vedou různé metody a postupy. Nebylo tomu jinak i u dodaných testovacích snímků, které obsahují větší množství šumu a nevýrazné objekty. V prvním kroku jsem se rozhodl odstranit šum pomocí Mediánového filtru (Kapitola 3.2). Na obrázku 14 lze vidět lékařský snímek po použití mediánového filtru. Mediánovému filtru byla nastavena velikost matice 5x5. Odzkoušeny byly i ostatní velikosti matic, avšak u matic menší než 5x5 zanechával filtr ve snímku stálé větší množství šumu a oproti tomu matice filtru větší než 5x5 zhoršovala kvalitu zájmového objektu.

Obrázek 14 - Použití mediánového filtru na lékařský snímek

Po odstranění šumu je zapotřebí více zvýraznit zájmové objekty a odstranit nezajímavé objekty. Jedním z postupů jak toho docílit, je použití metody prahování, kdy vhodně zvoleným prahem v mém případě šlo o automatické zjištění prahu z histogramu. Uživatel si proto sám nemohl zvolit práh, aby se předešlo nastavování prahu každému snímku či jednoho globálního prahu, kdy by však mohlo docházet ke špatně zvolené hodnotě, pokud byl rozdíl histogramu znatelný.



Obrázek 15 - Použití prahování na lékařském snímku

Registrace

Jak již bylo zmíněno v sekci Korekce, nejdůležitější části je zjistit posuv mezi dvěma částečně překrytými obrazy, a ten lze získat pomocí registrace obrazu. Pro registraci obrazu byly vytvořeny třídy *ImageCorrectionProcess* a *ImageRegistration*.

Třída *ImageRegistration*, jak její název napovídá, obstarává samotnou registraci a zjištění posuvu. Registrace se spustí zavoláním metody *Proceed(BufferedImage first, BufferedImage second)*, která přijímá jako parametry dva obrazy se snímky instance třídy *BufferedImage*, které následně předpřipraví odstraněním šumu (Mediánový filtr a prahování) zavoláním metody *ImagePrepare(BufferedImage img)* a provede Diskrétní dvourozměrnou Fourierovu transformaci.

Dalším krokem je Fázová korelace spuštěním metody *PhaseCorrelation(RenderedOp dftFirst, RenderedOp dftSecond),* kde se vkládají zmiňované Fourierovy transformace obrazů. Prvním úkonem této metody je komplexní sdružení na Fourierově transformaci druhého obrazu, následuje vynásobení s Fourierovou transformací prvního obrazu a podělení absolutní hodnotou daného výpočtu. Následně je proveden Gaussův filtr pro odstranění nevýznamných vrcholů a provedena inverzní Fourierova transformace. Po provedení inverzní Fourierovy transformace se vypočítá magnituda, která obsahuje bod registrace v rozích, proto se následně provede prohození kvadrantů, čímž je registrace umístěna v prostřední části obrazu. Nyní už zbývá jenom najít pozici nejvyššího vrcholu, čímž se získá místo

registrace a určí se posuv. Získaný posuv je použit jako návratová hodnota. S pomocí knihovny JAI byla implementace vcelku jednoduchá (Ukázka 3).

```
1.  private BufferedImage ImagePrepare(BufferedImage img, int threshold)
2.  {
3.      BufferedImage out = ImageExtension.Copy(img);
4.
5.      out = ImageExtension.MedianFilter(out, 5);
6.      out = ImageExtension.GrayScale(out);
7.      out = ImageExtension.Binarize(out, threshold);
8.      out = ImageExtension.GrayScale(out);
9.
10.     return out;
11. }
```

Ukázka 3 – Předzpracování obrazu pomocí knihovny JAI

Třída *ImageCorrectionProcess* dědí z třídy *SwingWorker* a implementuje rozhraní *Cancellable*. Pomocí třídy *SwingWorker* můžeme spouštět proces na pozadí a pomocí rozhraní *Cancellable* může proces kdykoliv zastavit. Běh složitých výpočtů na pozadí je v grafických aplikacích velice důležitý, protože tento úkol běží ve svém vlastním vlákně, nezablokuje se tak GUI, které se může dále překreslovat na základě uživatelských podnětů. Dále by se měly zobrazovat grafické prvky aplikace a provádět přístup ke GUI pouze z EDT. EDT je vlákno, jehož hlavním úkolem je zpracovávat události z AWT nebo Swingu. Pokud je zapotřebí aktualizovat GUI z nějakého jiného vlákna než EDT, docílíme toho pomocí metody *SwingUtili-ties.InvokeLater(Runnable)*. Třída *SwingWorker* obsahuje hlavní metody *doInBackground(), process(List<V> chunks), done()* a *execute()*. Do metody *doInBackground()* se vkládá kód jenž má běžet na pozadí. U dlouhých výpočetních úkonů, se může hodit zobrazovat aktuální stav procesu nebo kolik času ještě zbývá do konce. K tomuto účelu se využívá metoda *process(List<V> chunks)*. Tato metoda se volá z vlákna pomocí metody *publish(V... chunks)*, kde jako parametr můžeme vložit například číslo od nuly do sto, jenž by oznamoval aktuální stav zpracování v procentech. Pokud chceme po skončení metody *doInBackground()* provést ještě nějakou akci, použijeme metodu *done()*. Metoda *done()* a *process(List<V> chunks)*, již běží ve vlákně EDT, za pomocí nich lze upravovat GUI bez nutnosti provádět *Invoke*. Poslední zmiňovanou metodou je *execute*(), která spouští vlákno a tedy i výpočet.

Ve třídě *ImageCorrectionProcess* probíhá celý proces Kombinace obrazů. Třída přijímá skrze konstruktor *public ImageCorrectionProcess(Container container, ReferenceList list, CorrectionSetting setting)* parametry instance třídy *Container* obsahující data pro zpracování, list indexu referenčních obrazů a instanci třídy *CorrectionSetting* jenž obsahuje nastavení procesu. Z důvodu, že třída dědí z třídy *SwingWorker*, je veškerý výpočet prováděn v metodě *doInBackground()*. Celkový výpočet se liší podle parametrů, a to zda obsahuje referenční obrazy či má být použita průměrná hodnota.

Postup výpočtu je následující, v prvním kroku se zjistí, zda jsou pro zpracování obsaženy alespoň dva snímky, aby bylo možné výpočet provést. Není-li tomu tak, zpracování je ukončeno. Následuje uložení vybrané oblasti z nástroje *SelectionTool* do proměnné *selectionSquare*, která je instancí třídy *Rectangle2D*. Nebyl-li použit nástroj *SelectionTool* uloží se velikost snímku. Pokud v průvodci *SaveWizard* byla vybrána možnost vyplnění průměrnou barvou, inicializuje se pole o velikosti proměnné *selectionSquare*, které bude sloužit pro ukládání hodnot snímků pro finální výpočet průměrné hodnoty. Dále je vytvořena instance třídy *ImageRegistration* pro výpočet posuvu.

Další postup se dělí, na dvě časti. První částí je výpočet delty a nastavení jejího výpočtu. Snímky se prochází tak, že se vezme snímek s aktuálním indexem a následující snímek. Snímky se ořežou, pokud byl použit nástroj SelectionTool dle proměnné *selectionSquare* a následně jsou snímky vloženy do metody *Proceed(BufferedImage first, BufferedImage second)* třídy *ImageRegistration* čímž se získá posuv, který je uložen do listu *delties*. Za získaným posuvem může nastat uložení hodnot pro průměrnou hodnotu a to buď dojde k uložení hodnot obou snímků, pokud nebyl vybrán žádný referenční snímek nebo pouze hodnoty referenčního snímku má-li daný index.

Druhá část zkombinuje průměrný snímek a doplní chybějící části. Snímek je vybrán a ořezán jako v první sekci. Následně se vybere posuv pro daný snímek, pomocí kterého se snímek zarovná a chybějící části snímku jsou doplněny hodnotami průměrného snímku nebo uživatelsky definovanou barvou. Celý tento proces je proveden statickou metodou *Composite(BufferedImage backImg, BufferedImage foreImg, float deltaX, float deltaY, Rectangle2D window, Color color, AveragePixelArray array)* pomocné třídy *ImageExtension*. Metoda přijímá parametry, jakými jsou snímek určený ke korekci, snímek referenční, posun na ose X a ose Y, velikost výběru, a to buď celý obraz, nebo vybraná část, dále barvu výplně v případě zvolení vyplnění barvou a průměrné hodnoty. Pro zjištění chybějící části ve snímcích jsou snímky převedeny do stupňů šedi a porovnává se jejich hodnota jasu. Pokud je v průměrném či referenčním snímku hodnota jasu vyšší než ve snímku určeném ke korekci je tato část doplněna průměrnou hodnotou referenčních snímků nebo uživatelsky definovanou barvou, to záleží na zvoleném typu v průvodci *SaveWizard*. Takto upravený snímek je uložen do proměnné listu *temp*.

Uložení upravených snímků

Po provedení korekce snímku je následně uložen jako video nahrazením vadného snímku s vadným zájmovým objektem skrze třídu *VideoCreator* nebo jako obraz pomocí třídy *ImageSeqCreator*. Třídy mají implementované rozhraní *ICreator* obsahující metodu *public void Create(String filePath)*, přijímající cestu, kam se soubor uloží.

U volby uložení výsledku jako obrázek je snímek uložen s pomocí s pomocí knihovny javax.imageio.ImageIO, pokud se jedná o uložení do formátu BMP, PNG či GIF. Při zvolení formátu uložení jako JPEG je volána statická metoda *SaveAsJPEG(BufferedImage sourceImage, float quality, File path)* v pomocné třídě *ImageExtension*. Navíc u formátu JPEG lze nastavit i kvalitu v jaké má být obraz uložen, 100%, 75%, 50% nebo 25%.

Možnost uložení výsledku do videa je o něco komplikovanější než u možnosti uložení do sekvence snímků. U videa je nutné znát dobu zobrazení jednoho snímku. Určil jsem si rychlost o jeden snímek za sekundu více než v případě lidského oka, a to hodnotou 25. Při takovém počtu snímků za sekundu lze vidět obraz stále jako plynulý. Z počtu snímků za sekundu jsem provedl jednoduchý výpočet pro délku jednoho snímku, vydělil jsem jednu sekundu v milisekundách hodnotou počtem snímků za sekundu, čímž jsem dostal hodnotu 40 milisekund, po níž bude zobrazen jeden snímek ve videu. Snímky jsou postupně vybírány, konvertovány na barevné schéma BGR a vloženy do videa. Implementace uložení snímku do videa byla provedena pomocí knihovny Xuggler (Ukázka 4).

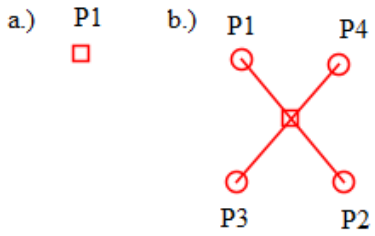
```
1.  @Override
2.  public void Create(String filePath)
3.  {
4.      filePath = filePath.replace("[I]", "");
5.
6.      final int videoStreamIndex = 0;
7.      final int videoStreamId = 0;
8.      long nextFrameTime = 0;
9.
10.     final IMediaWriter writer = ToolFactory.makeWriter(filePath);
11.     writer.addVideoStream(videoStreamIndex, videoStreamId, Width, Height);
12.
13.     int index = 0;
14.     while (nextFrameTime < Duration)
15.     {
16.         BufferedImage frame;
17.
18.         if(index <= Container.size() -1)
19.             frame = ImageExtension.convertToType(Container.get(index), BufferedImage
    .TYPE_3BYTE_BGR);
20.         else
21.             frame = ImageExtension.convertToType(Container.get(Container.size() -
     1), BufferedImage.TYPE_3BYTE_BGR);
22.
23.         writer.encodeVideo(videoStreamIndex, frame, nextFrameTime, DEFAULT_TIME_UNIT
    );
24.
25.         nextFrameTime += FRAME_RATE;
26.
27.         index++;
28.     }
29.
30.     writer.close();
31. }
```

Ukázka 4 - Uložení výsledků do Videa pomocí knihovny Xuggler

6.2 Korekce geometrických objektů na medicínském snímku

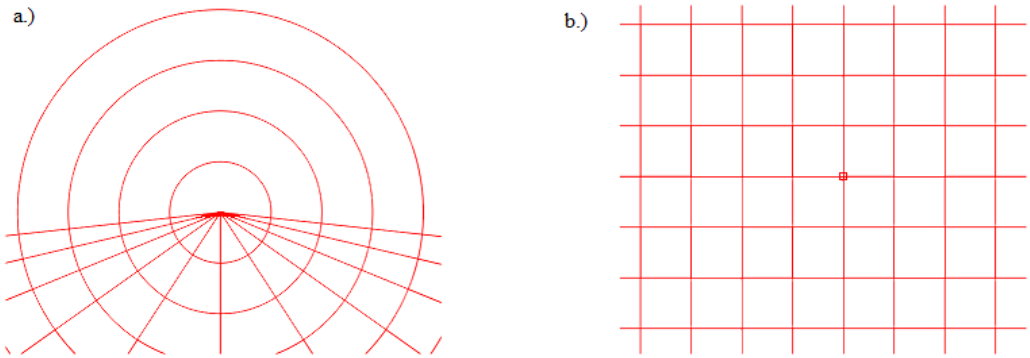
6.2.1 Analýza stávajících 2D objektů ve Fotomu NG

- **Bod** – Nejjednodušším 2D objektem v systému Fotom NG je prostý bod. Sledovaným parametrem je souřadnicová poloha bodu na zkoumaném snímku viz Obrázek 16 a).
- **Průsečík** – Objekt daný dvěma úsečkami. Sledovanými parametry jsou dva koncové body každé úsečky a průsečík úseček, viz Obrázek 16 b).



Obrázek 16 - a.) Bod, b.) Průsečík

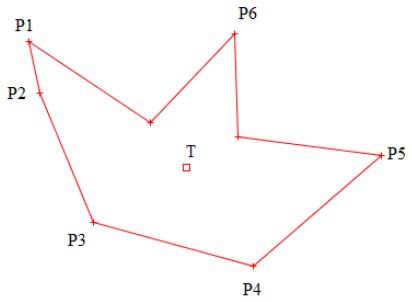
- **Mřížka** – Slouží jako jeden z nástrojů, který zobrazuje mřížku na snímku. Slouží pro ultrazvukové snímky a vytváří mřížku zobrazující soustředné kružnice a paprsky vycházející z definovaného místa. U mřížky sledujeme úhel, vzdálenost a souřadnicovou polohu středu, viz Obrázek 17 a).



Obrázek 17 – a.) Objekt mřížka b.) Objekt matice

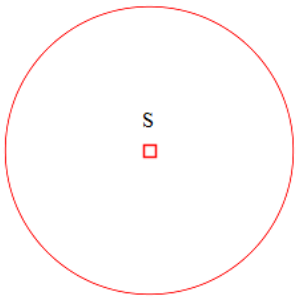
- **Matice** – Druhý z nástrojů pro zobrazení mřížky na snímku. Používá se pro ostatní typy snímků, viz Obrázek 17 b).

- **Polygon** – U polygonu sledujeme polohu těžiště, pozici bodu a plochu polygonu. Křivost určuje pokrytí křivek, které prokládají hrany polygonu, viz Obrázek 18.



Obrázek 18 - Objekt polygon

- **Kružnice** – Sledovanými parametry kružnice jsou poloha jejího středu, poloměr kružnice a plocha kružnice, viz Obrázek 19.



Obrázek 19 – Objekt Kružnice

Společným parametrem všech objektů je název a pozice jejich středu nebo těžiště v rámci zkoumaného snímku. Polygon je nejpoužívanější nástroj, kterým se definují oblasti, které nejsou nijak pravidelné. Oblast je možno definovat ručně, a to postupným definováním jednotlivých bodů nebo u rozsáhlých či velmi členitých objektů, je možno použít nástroj pro automatické definování objektu, který automaticky nalezne hranice objektu a definuje na něm polygon. Objekt kružnice se využije především u objektu kruhového tvaru, jakým je například tepna.

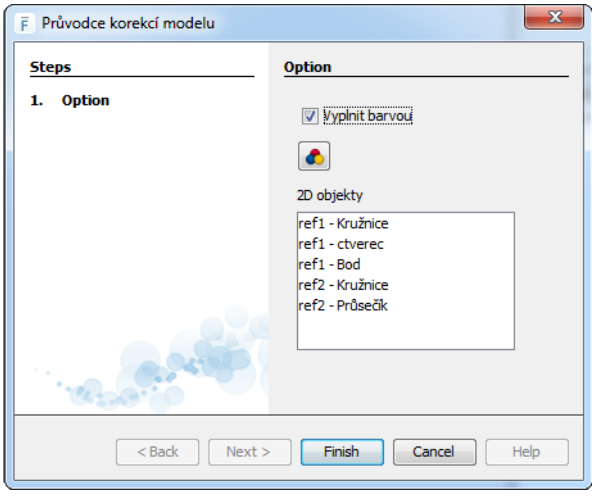
Všechny tyto objekty jsou definovány jako potomky třídy *ShapeTool*. Jedná se o abstraktní třídu odvozenou od obecné třídy *Tool*, která reprezentuje nástroje pro kreslení na plátno. *ShapeTool* vyhrazuje nástroje pro definování tvarů na snímku, a proto prakticky zosobňuje námi využívané zájmové objekty. Třídy objektů v sobě zahrnují spoustu předpřipravené funkčnosti, díky které můžeme velmi snadno získat souřadnice zájmového bodu nebo nastavit jednotlivé barvy pro události, kdy je objekt aktivní a neaktivní.

6.2.2 Specifikace požadavků

Druhou částí nového modulu pro systém Fotom NG byl požadavek na korekci 2D Fotom objektů ve snímku. Korekce má probíhat formou doplnění či úpravou stávajícího objektu ve snímku. Při úpravě objektů se pro korekci využívá objektů z jiných snímků, jenž mají stejné názvy. Přidané či upravené objekty bude možné rozlišit uživatelsky definovanou barvou.

6.2.3 Návrh GUI

Nyní, když už známe požadavky na druhou část modulu, je zapotřebí navrhnout, jaké ovládací prvky a možnosti využijeme a jak je začleníme do stávajícího GUI. Požadavkem na GUI je, aby bylo výsledné uživatelské rozhraní Fotomu NG integritní a využívalo stejných principů práce. Ze specifikace požadavků víme, že bude potřeba načíst objekty, podle kterých se následně upraví stávající objekty ve snímku nebo přidají objekty z referenčního snímku do aktuálního snímku. Tuto funkcionalitu obstará přidání nové možnosti volby do kontextového menu. Možnost bude aktivní, pokud bude vybrán více než jeden snímek v průzkumníku. Výběr barvy a objektů bude probíhat pomocí Netbeans Platform komponenty *Wizard* (Průvodce). Průvodce bude navržen podle vzhledu stávajících průvodců a bude obsahovat možnost vyplnit upravené či přidané objekty barvou a kdy při splnění této podmínky bude uživateli nabídnuta paleta barev. V průvodci bude také obsažen list názvů objektů s názvy snímků, ve kterých se daný objekt nachází, viz Obrázek 20.



Obrázek 20 - Průvodce korekcí modelů

6.2.4 Implementace

Položky kontextového menu jsou vytvářeny jako potomky třídy *NodeAction*. Takto vytvořené akce jsou volány událostmi nad Uzly (Nodes). Pro položku korekci objektu byla vytvořena třída *OpenFtmObjectsForCorrectionAction*, jenž dědí ze zmiňované třídy *NodeAction*. Třída *OpenFtmObjectsForCorrectionAction* obsahuje především funkcionalitu pro validaci a pokud je vybrán alespoň jeden snímek, který obsahuje 2D Fotom objekty, je možné provést korekci. Vytvořený potomek třídy *NodeAction* je nutné dále zaregistrovat. Registrace akce si vyžádala přidání kódu do již stávajícího modulu *fotom-explorer* a třídy *FtmObjectNode*. *Fotom-explorer* modul slouží k načítání snímků do systému Fotom a obstarává většinu práce v průzkumníku (Explorer). Kód byl přidán do metody *public Action[] getActions(boolean popup)*, jenž se volá pro zobrazení kontextové nabídky Uzlů (Nodes).

```
1.  ToolContainerI cont = Utilities.actionsGlobalContext().lookup(ToolContainerI.class);

2.  if(cont != null)
3.  {
4.      allActions.add(null);
5.      Lookup correctionActions = Lookups.forPath("Actions/FtmNodeCorrectionActions");
6.
7.      for (NodeAction action : correctionActions.lookupAll(NodeAction.class))
8.      {
9.          allActions.add(action);
10.     }
11. }
```

Ukázka 5 - Nová položka kontextové nabídky uzlu

Další krok probíhá pomocí vytvořeného průvodce *ModelCorrectionWizard*, jenž umožňuje uživateli zvolit, zda opravené a přidané objekty označit definovanou barvou a určit jaké objekty z referenčních snímků mají být zvoleny, viz obrázek 20.

Objekty referenčních snímků jsou pro korekci porovnány s objekty stávajícího snímku. Pokud název objektu není shodný s objektem stávajícího snímků, je objekt přidán do stávajícího snímku, není-li tomu tak, je objekt upraven. Úprava spočívá nastavením průměrné hodnoty vlastnosti 2D Fotom objektů, u kružnice se jedná o poloměr, u průsečíků průsečík a body přímek, všechny sledované vlastnosti objektu jsou popsány v kapitole 6.2.1. V kódu je ošetřen i stav, kdy dva referenční snímky obsahují různé objekty se stejným názvem tak, že se vybere pouze typ objektu s vyšší četností v referenčních snímcích. Vybral-li uživatel možnost doplnění přidaných a upravených objektů barvou označí se objety barvou, kterou uživatel zvolil v průvodci *ModelCorrectionWizard*. Označení objektů barvou je posledním krokem korekce objektu ve snímku.

Veškeré zpracování objektů snímků je uloženo ve třídě *ModelCorrection*.Metody *CorrectObjects(), GetMaxObjectInList(ArrayList<ShapeTool> tools)*, *ComputeObjetsAverage()*, *GetCenterFrmObject()* a *SetColor(ArrayList<ShapeTool> shapes, Color color)*, které slouží k nastavení korekce objektů, objektů s nejvyšším výskytem v listu, k výpočtu průměrné hodnoty objektu, získání středové pozice objektů a nastavení definované barvy objektům.

6.3 Testování

Kombinace obrazů a korekce objektů jako části modulu pro systém Fotom NG byly implementován v plném rozsahu specifikovaných požadavků na základě výše uvedené specifikace a analýzy. Použité postupy, algoritmy a jednotky kódu byly napříč stádii vývoje průběrně testovány, aby se dosáhlo potřebné kvality po stránce verifikační a validační. Testovací série dat se zaměřovala na otestování hlavní funkcionality. Testovací případy zasahovaly do těchto oblasti:

- načtení a uložení snímků
- určení posunu a spojení obrazů
- uživatelské rozhraní
- načtení
- korekce objektu
- vložení objektů do snímku

Testování probíhalo na testovacích snímcích a videích. Průměrná doba zpracování jednoho testovacího videa trvajícího 26 sekund a obsahující 671 snímků trvala jeden a půl hodiny při zpracování v jednom vlákně, což doba pro jedno zpracování dvou snímků trvala okolo osmi až devíti sekund. Zjištění velikosti posuvu mezi obrazy činila doba zpracování v průměru sedm až devět sekund. Z celkového času zpracování zabrala Fázová korelace v průměru 70%. Časy se týkají případů, kdy nebyl použit nástroj *SelectionTool*.

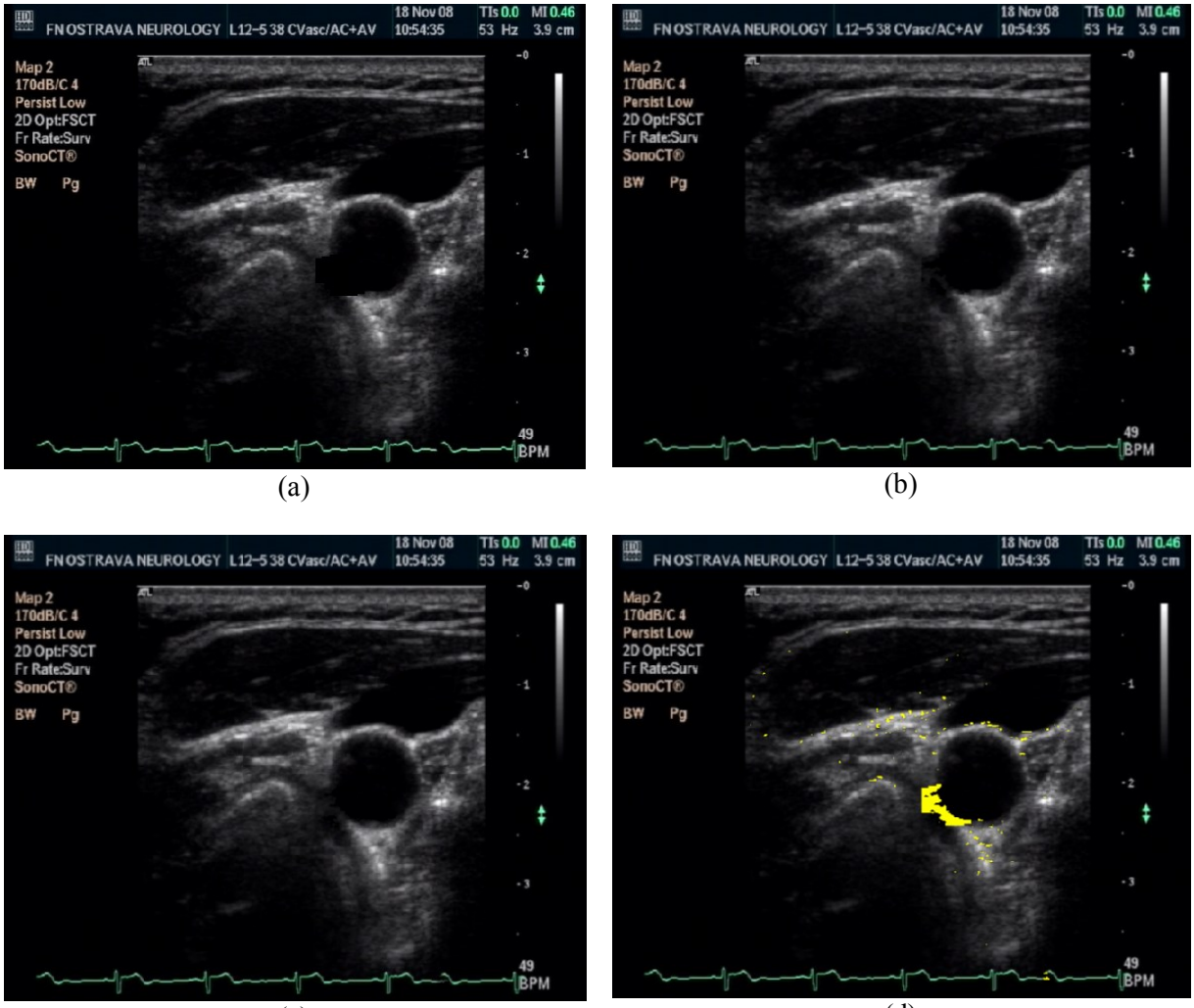
V tabulce 1 můžeme vidět způsob, jakým funguje korekce zájmového objektu ve snímku bez použití nástroje *SelectionTool*. Obrázek (a) v tabulce 1 je snímkem obsahující tepnu, která je v určité části nekorektní a chybí část její stěny. Pro korekci této tepny byly vybrány referenční snímky podobné obrazu snímku (c) a snímky obsahovaly určitý posun vůči poškozenému snímku, nebyl-li posuv a deformace zájmového objektu příliš velká byla korekce chybějící části tepny ve snímku provedena úspěšně.

Obrázek (c) snímku je výsledek korekce obrázku (a) snímku po doplnění průměrnými hodnotami barev získaných z referenčních snímků. V obrázku (b) snímku byl použit referenční snímek se stejnou vadou jako obrázek (a) snímku a snímek bez vady tepny. Tyto obrázky při zprůměrování hodnot barev vytvořily snímek s poloviční hodnotou v místě chybějící části tepny.

Poslední obrázek snímku v tabulce 1 je obrázek (d) snímku u nějž byly použity referenční snímky pro vytvoření průměrné hodnoty a následně v místě snímku s chybějící části stěny tepny, kdy rozdíl barev byl vetší než hodnota 5, byl doplněn uživatelsky zvolenou barvou skrze průvodce *SaveWizard* v tomto případě žlutou, která je také defaultní barvou při vybrání možnosti doplnění chybějící části snímku barvou.

Kvalita a úspěšnost provedení korekce záleží především na vhodně zvolených referenčních snímků. Rozdíl intenzity nebo rozdíl histogramu referenčního snímku a vadného snímku by měl být co nejmenší, aby došlo ke správnému určení místa chybějící části snímku, které má být doplněno průměrnou hodnotou nebo uživatelsky definovanou barvou. Dalším důležitým faktorem je míra deformace snímku, kdy zájmový objekt je deformace tak, že přesahuje svou vlastní hranici objektu

v korektním stavu, což může vést k doplnění i částí, jenž jsou správné a nevyžadují korekci. Tímto pak dochází ke špatné kvalitě korekci a korekci lze označit jako neúspěšnou.



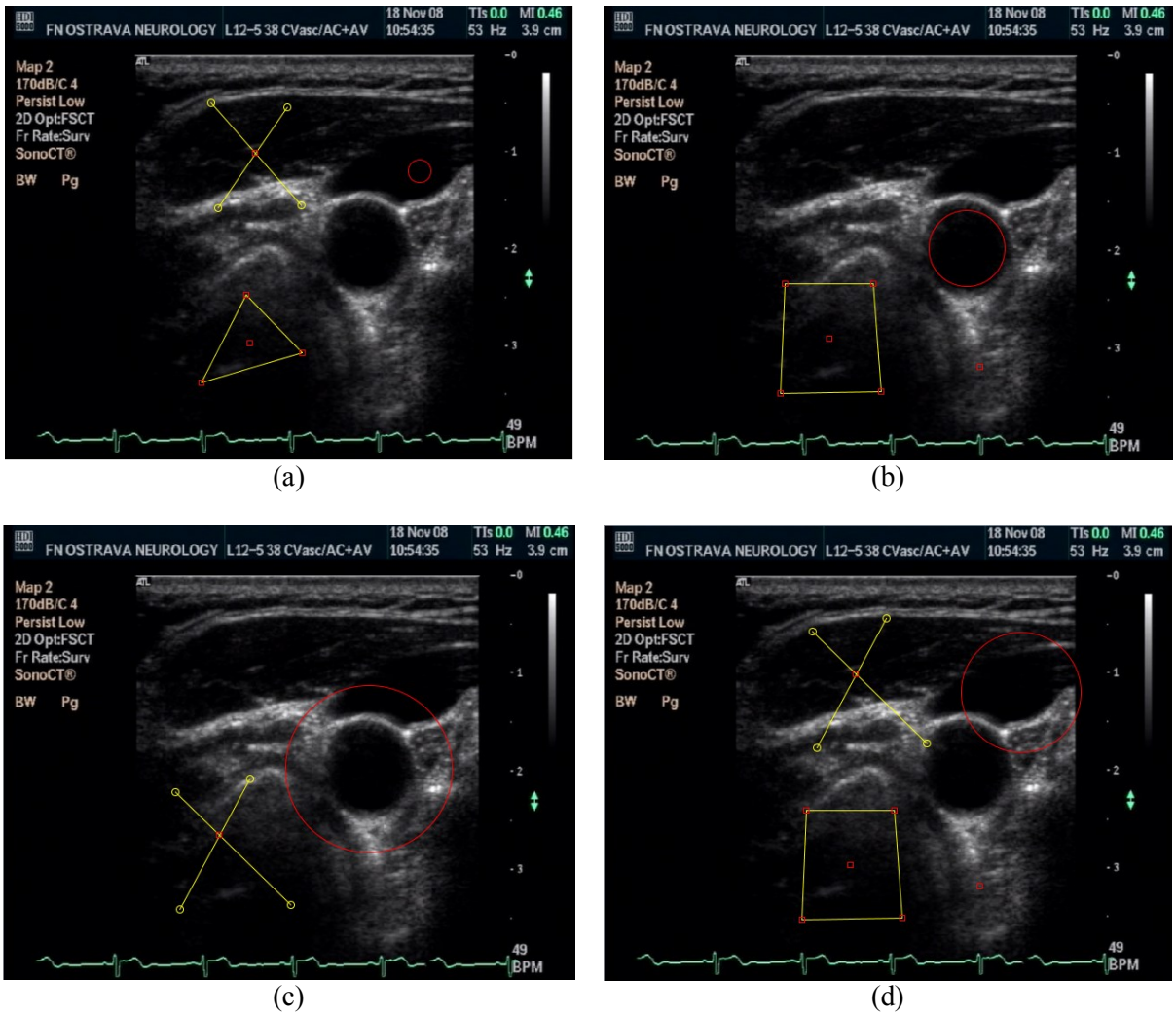
(a) (b)

(c) (d)

Tabulka 1 – a) Snímek s chybějící části tepny, b) Opravená část tepny, při použití referenčního snímku a snímku s chybějící části tepny pro korekci, c) Opravená část tepny, doplněna průměrnou hodnotou z hodnot referenčních snímků d) Opravena část tepny, doplněna uživatelsky definovanou barvou.

Druhá část testování probíhala na snímcích obsahující objekty systému Fotom jenž měli špatně definované své vlastnosti nebo snímek neobsahoval požadované objekty. Tyto objekty byly doplněny či upraveny dle objektů z referenčních snímků. Tabulka 2, znázorňuje průběh jednoho testování této časti modulu. Obrázek (a) snímku obsahuje tři nekorektní Fotom objekty ty jsou následně opraveny pomocí objektů v obrazech (b) a (c) snímků. Výsledek po provedení korekce obrázku (a) snímku je obrázek (d) snímku. Výsledný snímek obsahuje navíc objekt bod z obrázku (b) snímku, z kterého byla provedena i korekce objektu polygon, který byl z trojúhelníku převeden na čtverec. Z obrázku (c)

snímku proběhla korekce průsečíku úseček a při použití objektu kruhu z obrazu (b) snímku se opravil původní kruh doplněním průměrné hodnoty poloměru kruhu z referenčních snímků.



Tabulka 2 - a) Snímek s objekty určenými pro korekci, b) Referenční snímek číslo 1, c) Referenční snímek číslo 2, d) Opravené objekty ve snímku při použití objektů z referenčních snímků číslo 1 a 2.

Všechny definované testovací případy uvedeny na začátku této kapitoly byly spuštěny, dokončeny a proběhly úspěšně. Otestována byla také integrace se systémem Fotom NG. Navrhovaný modul je připraven na běžný provoz, avšak je zde stále riziko drobných chyb nebo nedostatků, které budou odstraňovány v průběhu nasazení.

7. Závěr

Během tvorby diplomové práce jsem se seznámil důkladně s metodami pro zpracování digitálního obrazu, především v metodice zjišťování podobností mezi dvěma obrazy a předzpracování obrazu za účelem zjištění posuvu objektů na ultrazvukových snímcích. Využil jsem znalostí získaných z předmětů zabývajícími se zpracováním digitálního obrazu vyučované na Fakultě elektrotechniky a informatiky Vysoké školy báňské - Technické univerzity Ostrava.

Dále jsem se seznámil s vědním oborem fotogrammetrie, abych pochopil její způsoby a užívanou terminologii. Získal jsem vědomosti o způsobech pořizování medicínských snímků, o jejich problémech a jak je odstranit. Tyto informace jsem využil při zpracování diplomové práce. Dále jsem se seznámil se systémem Fotom NG jako nástrojem pro zpracování digitálních lékařských či důlních snímků.

Cílem mé diplomové práce bylo vyvinout modul pro korekci objektů ve snímku a kombinaci snímků za účelem vytvoření jednoho kvalitního snímku. Snímky s novým modulem lze zpracovat, jak z obrazů, tak i z videa. Modul byl implementován do systému Fotom NG pod názvem *fotom-image-correction*. Specifikovány, navrhnuty, implementovány a otestovány byl veškeré požadavky zadané vedoucím práce. V průběhu specifikace se na základě konzultací s vedoucím práce navrhlo několik vylepšení a drobných úprav.

Analýzou stávajících modulů systému Fotom NG jsem získal základní přehled o tom, jaký by měl mít nový modul vzhled a funkčnost. Určil jsem jeho slabá místa a těm se nažil vyhnout. Při implementaci modulu jsem využil svých zkušeností s programovacím jazykem Java. S frameworkem Netbeans platform jsem při tvorbě této práce přišel poprvé do styku. Bylo tedy potřeba se naučit s tímto frameworkem pracovat a získat potřebné zkušenosti pro jeho použití při tvorbě modulu do systému Fotom NG. Nabité zkušenosti mi také pomohlo pochopit celkové fungování a provázanost systému Fotom NG a jeho API. Při tvorbě videa jsem se rozhodl využít knihovny xuggler jenž již v systému byla implementována a nedocházelo k zbytečnému navyšování velikosti systému Fotom NG. Knihovna Xuggler pokryla dostatečně požadavky na zpracování videa nutného pro možné zpracování.

Veškeré vědomosti, které jsem získal a využil, jsem shrnul v této práci. Diplomová práce je popsána od obecných částí až po detailní popis užitých metod a postupů. Celá práce tak popisuje postup tvorby a sled seznamování s danou problematikou. jenž byly použity.

V návrhu, implementaci a testování jsem rozdělil na tři části pro lepší přehlednost a logické oddělení, popsal nejdůležitější třídy modulů a definované API. Řešení jsem ilustroval pomocí vhodných diagramů. Hotové dílo jsem otestoval dle testovacích případů sestavených na základě specifikace. Zaměřil jsem se na hlavní funkčnost a praktickou použitelnost. Testováním jsem potvrdil funkčnost jednotlivých částí modulu, jeho přesnost a důsledky integrace. Na základě zmíněných testů jsem rozhodl, že je nový modul dostatečně připraven pro nasazení, avšak připustil jsem možnost drobných chyb, které se mohou během reálného užívání vyskytnout.

Nový modul byl navržen jako snadno upravitelná a rozšiřitelná komponenta systému Fotom NG. Vyskytnou-li se nové požadavky nebo požadavky na změny lze modul jednoduše rozšířit a vylepšit. Součástí diplomové práce je jak uživatelská, tak programátorská dokumentace, kterou jsem vypracoval a přiložil k této práci. S novým modulem lze doplňovat chybějící částí ve snímcích a upravovat objekty, což pomůže například ke sledování stavu pacienta v čase.

Fotom NG je jedinečný a ucelený nástroj pro práci s lékařskými a důlními snímky. V současné době jsou do systému FOTOM NG vyvíjeny další dva moduly. První vyvíjí Jan Tomeček a jeho obecný modul pro nalezení objektu ve snímku. Autorem druhého modulu je Tomáš Hudeček, jenž hledá také objekty ve snímku, avšak pomocí metod aktivních kontur a level-setů.

Literatura

[1] Střední průmyslová škola zeměměřická, *Kapitoly z fotogrammetrie* [online], Dostupné z: http://www.spszem.cz/index.php/vyuka/povinne-pedmty/fotogrammetrie.html [cit. 2013-03-10]

[2] SZELISKI R., *Image Alignment and Stitching: A Tutorial* [online], 2004 Dostupné z: <http://research.microsoft.com/pubs/70092/tr-2004-92.pdf> [cit. 2013-03-10]

[3] DLUHOŠ P., *Metaheuristické optimalizační metody pro registraci obrazů z magnetické rezonance algoritmech* [online], (2011), Bakalářská práce, Dostupné z: <http://is.muni.cz/th/269281/prif_b_b1/bakalarskaPrace.pdf>

[4] LIČEV L., *Systém FOTOM 2008 a vizualizace procesu měření* In: Sborník z GIS Ostrava 2009, Ostrava 25.-28.1.2009. Dostupné z <http://gis.vsb.cz/GIS_Ostrava/GIS_Ova_2009/sbornik/index.htm>, ISBN 978-80-87294-00-0,

[5] BÖCK H., *Platforma NetBeans : Podrobný průvodce programátora*. Vydání první. Brno : Computer Press, a.s., 2010. 320 s. ISBN 978-80-251-3116-9

[6] ZAJÍC Petr, *Analýza videozáznamů ultrazvukových vyšetření*, diplomová práce, 2011.

[7] HRAZDIRA I., *ÚVOD DO ULTRASONOGRAFIE* [online], Dostupné z: <http://www.med.muni.cz/dokumenty/pdf/uvod_do_ultrasonografie1.pdf> [cit. 2013-03-20]

[8] KRAHULEC L., *Počítačové zpracování fotografie,* diplomová práce, 2009.

[9] Technická univerzita v Liberci, Fakulta Textilní, *Úvod do zpracování a analýzy obrazu* [online], Dostupné z: <https://blade1.ft.tul.cz/~tyr/htdocs/elearning/Media/File/5/123/P1.pdf> [cit. 2013-03-25]

[10] FIŘT J., HOLOTA R., *Digitalizace a zpracování obrazu* [online], 2008 Dostupné z: <http://home.zcu.cz/~holota5/publ/DigZprO.pdf> [cit. 2013-03-26]

[11] HOLČÍK J., *Analýza biologických a klinických dat v mezioborovém pojetí,* Vzorkování [online]. Dostupné z: < http://www.iba.muni.cz/summer-school2009/res/file/holcik-vzorkovani.pdf> [cit. 2013-03-28]

[12] ZIKMUND T., *Měření rychlosti objektů pomocí Fourierovy transformace*. Brno, 2008, Diplomová práce na fakultě strojního inženýrství Vysoké Učení Technické, Ústav Matematiky. Vedoucí diplomové práce Ing. PAVEL ŠTARHA, Ph.D.

[13] SOJKA E., *Digitální zpracování a analýza obrazu*, Ostrava 2000. ISBN: 80-7078-746-5

[14] MIKŠÍK O., *Praktické využití metod digitálního zpracování obrazu*[online], 2007 Dostupné z: <http://soc.nidm.cz/data/2007/01-2.pdf> [cit. 2013-04-02]

[15] SHEKARFOROUSH H., BERTHOD M., ZERUBIA J. *Subpixel Image Registration by Estimating the Polyphase Decomposition of the Cross Power Spectrum*. Proceedings 1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, strany 532 - 537, červen 1996, USA [online]. Dostupné z: <http://cil.cs.ucf.edu/pdf/foroosh_cvpr96_polyphase.pdf> [cit. 2013-04-06]

[16] *Wikipédie*, Frame rate [online], Dostupné z: <http://en.wikipedia.org/wiki/Frame_rate> [cit. 2013-04-10]

[17] Krčmář M., *Spojování obrazů* [online], Dostupné z: <http://cmp.felk.cvut.cz/cmp/courses/pvi2000/Labs/Ulohy/krcmar.1/1.htm> [cit. 2013-04-22]

Přílohy na CD

- Zdrojové kódy
- Spustitelná distribuce programu
- Uživatelská příručka
- Programátorská příručka